Programación anual del módulo profesional

Programación

Ricardo Pérez López

Curso 2020/2021

Departamento de Informática y Comunicaciones
Jefe de Departamento: Ricardo Pérez López

Índice

1. Información general

- **Normativa de aplicación:** [1], [2]
- **Equivalencia en créditos ECTS:** 14.
- **Código:** 0485.
- **Duración total:** 256 horas (32 semanas)
- **Carga lectiva semanal:** 8 horas

2. Objetivos generales

1. La formación del módulo contribuye a alcanzar los **objetivos generales** de este ciclo formativo que se relacionan a continuación [1]:
   - *e)* Interpretar el diseño lógico, verificando los parámetros establecidos para gestionar bases de datos.
   - *j)* Emplear herramientas y lenguajes específicos, siguiendo las especificaciones, para desarrollar componentes multimedia.
   - *q)* Programar y realizar actividades para gestionar el mantenimiento de los recursos informáticos.
2. La formación del módulo contribuye a alcanzar las **competencias profesionales, personales y sociales** de este título que se relacionan a continuación [1]:
   - *a)* Configurar y explotar sistemas informáticos, adaptando la configuración lógica del sistema según las necesidades de uso y los criterios establecidos.
   - *e)* Desarrollar aplicaciones Web con acceso a bases de datos utilizando lenguajes, objetos de acceso y herramientas de mapeo adecuados a las especificaciones.
   - *f)* Integrar contenidos en la lógica de una aplicación Web, desarrollando componentes de acceso a datos adecuados a las especificaciones.
   - *i)* Integrar componentes multimedia en el interface de una aplicación Web, realizando el análisis de interactividad, accesibilidad y usabilidad de la aplicación.
   - *j)* Desarrollar e integrar componentes software en el entorno del servidor Web, empleando herramientas y lenguajes específicos, para cumplir las especificaciones de la aplicación.
   - *v)* Realizar la gestión básica para la creación y funcionamiento de una pequeña empresa y tener iniciativa en su actividad profesional con sentido de la responsabilidad social.

3. Resultados de aprendizaje y criterios de evaluación

Los resultados de aprendizaje del módulo y sus criterios de evaluación asociados son los que se describen a continuación [1]:

[RA1] Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

Criterios de evaluación:

- *CE1.a)* Se han identificado los bloques que componen la estructura de un programa informático.
- *CE1.b)* Se han creado proyectos de desarrollo de aplicaciones.
- *CE1.c)* Se han utilizado entornos integrados de desarrollo.
- *CE1.d)* Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.
- *CE1.e)* Se ha modificado el código de un programa para crear y utilizar variables.
- *CE1.f)* Se han creado y utilizado constantes y literales.
- *CE1.g)* Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
- *CE1.h)* Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
- *CE1.i)* Se han introducido comentarios en el código.

[RA2] Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

Criterios de evaluación:

- *CE2.a)* Se han identificado los fundamentos de la programación orientada a objetos.
- *CE2.b)* Se han escrito programas simples.
- *CE2.c)* Se han instanciado objetos a partir de clases predefinidas.
- *CE2.d)* Se han utilizado métodos y propiedades de los objetos.
- *CE2.e)* Se han escrito llamadas a métodos estáticos.
- *CE2.f)* Se han utilizado parámetros en la llamada a métodos.
- *CE2.g)* Se han incorporado y utilizado librerías de objetos.
- *CE2.h)* Se han utilizado constructores.

CE2.i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.

[RA3] Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.

Criterios de evaluación:

CE3.a) Se ha escrito y probado código que haga uso de estructuras de selección.

CE3.b) Se han utilizado estructuras de repetición.

CE3.c) Se han reconocido las posibilidades de las sentencias de salto.

CE3.d) Se ha escrito código utilizando control de excepciones.

CE3.e) Se han creado programas ejecutables utilizando diferentes estructuras de control.

CE3.f) Se han probado y depurado los programas.

CE3.g) Se ha comentado y documentado el código.

[RA4] Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

Criterios de evaluación:

CE4.a) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.

CE4.b) Se han definido clases.

CE4.c) Se han definido propiedades y métodos.

CE4.d) Se han creado constructores.

CE4.e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

CE4.f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

CE4.g) Se han definido y utilizado clases heredadas.

CE4.h) Se han creado y utilizado métodos estáticos.

CE4.i) Se han definido y utilizado interfaces.

CE4.j) Se han creado y utilizado conjuntos y librerías de clases.

[RA5] Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

Criterios de evaluación:

CE5.a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

CE5.b) Se han aplicado formatos en la visualización de la información.

CE5.c) Se han reconocido las posibilidades de entrada / salida del lenguaje y las librerías asociadas.

CE5.d) Se han utilizado ficheros para almacenar y recuperar información.

CE5.e) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

CE5.f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

CE5.g) Se han programado controladores de eventos.

CE5.h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

[RA6] Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

Criterios de evaluación:

CE6.a) Se han escrito programas que utilicen arrays

CE6.b) Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.

CE6.c) Se han utilizado listas para almacenar y procesar información.

CE6.d) Se han utilizado iteradores para recorrer los elementos de las listas.

CE6.e) Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles.

CE6.f) Se han creado clases y métodos genéricos.

CE6.g) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.

CE6.h) Se han identificado las clases relacionadas con el tratamiento de documentos XML.

CE6.i) Se han realizado programas que realicen manipulaciones sobre documentos XML.

[RA7] Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

Criterios de evaluación:

CE7.a) Se han identificado los conceptos de herencia, superclase y subclase.

CE7.b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

CE7.c) Se ha reconocido la incidencia de los constructores en la herencia.

CE7.d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

CE7.e) Se han diseñado y aplicado jerarquías de clases.

CE7.f) Se han probado y depurado las jerarquías de clases.

CE7.g) Se han realizado programas que implementen y utilicen jerarquías de clases.

CE7.h) Se ha comentado y documentado el código.

[RA8] Utiliza bases de datos orientadas a objetos, analizando sus características y aplicando técnicas para mantener la persistencia de la información.

Criterios de evaluación:

CE8.a) Se han identificado las características de las bases de datos orientadas a objetos.

CE8.b) Se ha analizado su aplicación en el desarrollo de aplicaciones mediante lenguajes orientados a objetos.

CE8.c) Se han instalado sistemas gestores de bases de datos orientados a objetos.

CE8.d) Se han clasificado y analizado los distintos métodos soportados por los sistemas gestores para la gestión de la información almacenada.

CE8.e) Se han creado bases de datos y las estructuras necesarias para el almacenamiento de objetos.

CE8.f) Se han programado aplicaciones que almacenen objetos en las bases de datos creadas.

CE8.g) Se han realizado programas para recuperar, actualizar y eliminar objetos de las bases de datos.

CE8.h) Se han realizado programas para almacenar y gestionar tipos de datos estructurados, compuestos y relacionados.

[RA9] Gestiona información almacenada en bases de datos relacionales manteniendo la integridad y consistencia de los datos.

Criterios de evaluación:

CE9.a) Se han identificado las características y métodos de acceso a sistemas gestores de bases de datos relacionales.

CE9.b) Se han programado conexiones con bases de datos.

CE9.c) Se ha escrito código para almacenar información en bases de datos.

CE9.d) Se han creado programas para recuperar y mostrar información almacenada en bases de datos.

CE9.e) Se han efectuado borrados y modificaciones sobre la información almacenada.

CE9.f) Se han creado aplicaciones que ejecuten consultas sobre bases de datos.

CE9.g) Se han creado aplicaciones para posibilitar la gestión de información presente en bases de datos relacionales.

4. Instrumentos y procedimientos de evaluación y calificación

La evaluación tendrá como finalidad determinar el nivel de competencia de los alumnos y la consecución de los objetivos. Se desarrollará de forma continua, y atenderá a los siguientes aspectos:

- Aprendizaje autónomo, viendo la capacidad del alumno para interiorizar, gestionar y participar en los procesos de aprendizaje propios.
- Comprensión del lenguaje común.
- Adquisición de conceptos básicos del módulo profesional que permiten al alumno incluirlos como un elemento más de su realidad profesional.
- Participación y trabajo en grupo, viendo la capacidad que tiene este de escuchar y debatir las diferentes soluciones de un problema.
- Nivel de abstracción alcanzado.

Para que el seguimiento de dicha evaluación sea factible, el alumno deberá asistir con regularidad a las clases y participar activamente en las mismas, de forma que una sistemática y frecuente falta de asistencia a clase supondrá para el alumno la **pérdida de la evaluación continua** y sólo tendrá derecho a un examen final. Asimismo, se requiere que el alumno **acceda al menos diariamente a las plataformas iPasen y Moodle Centros** y que **revise diariamente su correo en el dominio `@iesdonana.org`** para informarse puntualmente de las novedades que pudieran darse en el módulo, quedando claro que **es responsabilidad del alumno informarse activamente sobre las mismas**.

4.1. Valoración general de los contenidos

Los contenidos se ponderarán en base a los siguientes porcentajes:

Trabajos, actividades y ejercicios (casa, clase, grupo) (**TRA**)	30 %
Pruebas evaluativas (**EXA**)	70 %

Si, por algún motivo, no se pudiera evaluar uno de los dos apartados anteriores (**TRA** o **EXA**), el otro apartado restante soportaría el 100 % de la carga evaluativa, de forma que la calificación final resultaría únicamente de dicho apartado.

La evaluación de cualquiera de los trabajos, actividades, ejercicios y pruebas evaluativas podrá requerir, a criterio y discreción del profesor del módulo, la defensa de los mismos en una entrevista individual con el alumno para garantizar la autoría y la adquisición adecuada de las competencias correspondientes.

4.2. Calificación

Se llevarán a cabo trabajos, actividades y/o ejercicios (apartado **TRA**) versados sobre los contenidos trabajados en una unidad didáctica o bloque de unidades didácticas conceptualmente relacionadas. (Esto significa, en consecuencia, que no es obligatoria la realización de trabajos, actividades y/o ejercicios en cada unidad didáctica, sino que a tales efectos se pueden agrupar varias unidades didácticas.)

Dentro de este grupo podrá incluirse alguna prueba (tipo test o de respuestas cortas) a responder de forma individual sobre conocimientos teóricos de determinados aspectos básicos asociados a unidades (o conjunto de unidades) didácticas concretas.

Asimismo, se realizará un examen al final de cada evaluación parcial (apartado **EXA**), coincidiendo aproximadamente con el final del trimestre correspondiente. Debido al carácter de evaluación continua del módulo, así como del hecho de que cada contenido trabajado se asienta sobre los anteriores, es posible que el alumno requiera, para la evaluación positiva de un trimestre, el conocimiento necesario de contenidos de trimestres anteriores, independientemente de que dichos conocimientos hayan sido evaluados positiva o negativamente en trimestres anteriores.

Cada examen, a su vez, constará de dos partes bien diferenciadas:

Parte teórica: Una prueba que versará sobre conocimientos esenciales con preguntas tipo test, de respuestas cortas o similar (apartado **TEO**).

Parte práctica: Una prueba práctica con problemas y ejercicios donde el alumno deberá aplicar lo aprendido escribiendo código real (apartado **PRA**).

La nota del examen (apartado **EXA**) se calculará como una media ponderada de ambas partes **TEO** y **PRA** y para superar el examen en su conjunto será necesario obtener una calificación mínima en ambas partes, según se recoge en el siguiente epígrafe.

Asimismo, para superar una evaluación parcial será necesario obtener una calificación mínima tanto en el apartado **TRA** como en el apartado **EXA** de dicha evaluación parcial, según se recoge en el siguiente epígrafe.

4.2.1. Calificaciones parciales

La **calificación de cada evaluación parcial** (primer, segundo y tercer trimestres por separado) se calculará de la siguiente forma:

Algoritmo 1 (Cálculo de la calificación parcial)

```
if TEO ≥ 3 and PRA ≥ 4:
    EXA = TEO * 0.3 + PRA * 0.7
else:
    EXA = min(TEO, PRA)

if TRA ≥ 3 and EXA ≥ 4:
    NOTA = TRA * 0.3 + EXA * 0.7
else:
    NOTA = min(TRA, EXA)
```

Donde:

TRA: Media aritmética de las calificaciones de los trabajos realizados en ese trimestre, valorados del 0 al 10.

TEO: Calificación de la parte teórica del examen correspondiente a ese trimestre, valorada del 0 al 10.

PRA: Calificación de la parte práctica del examen correspondiente a ese trimestre, valorada del 0 al 10.

EXA: Calificación del examen correspondiente a ese trimestre, valorada del 0 al 10, calculada como la media ponderada de **TEO** y **PRA**.

La evaluación parcial se considera aprobada si **NOTA** $\geq$ 4,5.

Si durante el trimestre en cuestión no se realizaran trabajos, actividades y/o ejercicios (apartado **TRA**) dignos de calificación, entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial correspondiente coincidirá con la nota del examen (apartado **EXA**), por lo que quedará simplemente de la siguiente forma:

$$\mathbf{NOTA} = \mathbf{EXA}$$

Asimismo, si durante el trimestre en cuestión no se realizaran exámenes (apartado **EXA**), entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial correspondiente coincidirá con la nota de los trabajos, actividades y/o ejercicios (apartado **TRA**), por lo que quedará simplemente de la siguiente forma:

$$\textbf{NOTA} = \textbf{TRA}$$

Las **faltas de ortografía** en los exámenes serán **penalizadas** según acuerdo del Departamento, de la siguiente forma:

Número de faltas	Evaluación 1	Evaluación 2
< 5	$-0,25$ puntos	$-0,50$ puntos
≥ 5	$-0,50$ puntos	$-1,00$ puntos

En ningún caso este criterio podrá ser motivo de que el alumno no supere la prueba escrita.

4.2.2. Calificación final

Para superar el módulo será necesario obtener una calificación mínima en cada evaluación parcial, según se establece en el siguiente algoritmo, donde se determina cómo se calcula la calificación final del módulo:

Algoritmo 2 (Cálculo de la calificación final)

```
if min(EV₁, EV₂, EV₃) ≥ 4:
    NOTA = EV₁ * 0.33 + EV₂ * 0.33 + EV₃ * 0.33
else:
    NOTA = min(EV₁, EV₂, EV₃)
```

Donde:

- $\textbf{EV}_1$: Calificación de la primera evaluación.
- $\textbf{EV}_2$: Calificación de la segunda evaluación.
- $\textbf{EV}_3$: Calificación de la tercera evaluación.

El módulo se considera aprobado si $\textbf{NOTA} \geq 4{,}5$.

4.3. Medidas de recuperación y mejora de las calificaciones

4.3.1. Recuperación del apartado TRA

A lo largo del curso se abrirán nuevas ventanas temporales para que el alumnado entregue las correcciones necesarias en aquellas actividades que estén pendientes de evaluación positiva. La evaluación de tales correcciones podrá requerir, a criterio y discreción del profesor del módulo, la defensa de las mismas en una entrevista individual con el alumno para garantizar la autoría y la adquisición adecuada de las competencias correspondientes.

4.3.2. Recuperación del apartado EXA

Al final de cada evaluación, o a comienzos de la evaluación siguiente, se llevará a cabo una prueba de recuperación para aquellos alumnos que tengan algún contenido pendiente en esa evaluación. Al final del curso se hará una prueba final de recuperación para aquellos alumnos que tengan pendientes contenidos de alguna evaluación, debiendo presentarse únicamente a aquellas partes que tengan pendientes. La evaluación de tales pruebas de recuperación podrá requerir, a criterio y discreción del profesor del módulo, la defensa de las mismas en una entrevista individual con el alumno para garantizar la autoría y la adquisición adecuada de las competencias correspondientes.

4.3.3. Mejora de las calificaciones

El alumno con contenido superado que quieran mejorar sus calificaciones («subir nota») podrá, si así lo desea, someterse al protocolo de recuperación y realizar de nuevo las actividades del apartado **TRA** que desee, o bien las pruebas de recuperación del apartado **EXA** que desee (o bien ambas cosas) pero, en tal caso, **es consciente, entiende, asume y acepta que la calificación que finalmente constará en dicha actividad o prueba será la alcanzada en ese último intento, por lo que corre el riesgo de acabar con menos nota que antes**.

5. Contenidos y temporalización

Cada unidad didáctica tiene una duración temporal de **una semana**, que podrá ampliarse o reducirse en función de las circunstancias cuando el profesor lo estime conveniente (por ejemplo, para la realización de actividades, ejercicios y prácticas en clase).

Los contenidos marcados con la etiqueta *#opcional* son contenidos complementarios que sólo se impartirán si hay tiempo suficiente para ello y nunca a costa de otros contenidos no

opcionales. También podrán ser usados como elementos a desarrollar para el alumnado con altas capacidades o que manifieste un ritmo de aprendizaje superior al del resto del grupo/clase.

Todas las fechas se muestran en formato ISO 8601 (*año-mes-día*).

5.1. Cuadro resumen

Unidad didáctica	Inicio estimado
1. Introducción *#ev1*	2020-09-21
2. Expresiones *#ev1*	2020-09-28
3. Programación funcional (I) *#ev1*	2020-10-05
4. Programación funcional (II) *#ev1*	2020-10-12
5. Programación imperativa *#ev1*	2020-10-19
6. Programación estructurada *#ev1*	2020-10-26
7. Tipos de datos estructurados *#ev1*	2020-11-02
8. Programación modular (I) *#ev1*	2020-11-09
9. Calidad (I) *#ev1*	2020-11-16
10. Abstracción de datos *#ev1*	2020-11-23
11. Programación orientada a objetos *#ev2*	2021-01-11
12. Relaciones entre clases *#ev2*	2021-01-18
13. Introducción a la tecnología Java *#ev2*	2021-01-25
14. Elementos básicos del lenguaje Java *#ev2*	2021-02-01
15. Programación orientada a objetos en Java *#ev2*	2021-02-08
16. Diseño de clases en Java	2021-02-15
17. Relaciones entre clases en Java *#ev2*	2021-02-22
18. Programación modular (II) *#ev2*	2021-03-02
19. Programación genérica *#ev2*	2021-03-09
20. Estructuras de datos lineales *#ev2*	2021-03-16
21. Ordenación y búsqueda *#ev3*	2021-04-05
22. Estructuras de datos no lineales *#ev3*	2021-04-12
23. Control de excepciones en Java *#ev3*	2021-04-19
24. Java Collections Framework *#ev3*	2021-04-26
25. Calidad (II) *#ev3*	2021-05-04
26. Gestión de bases de datos relacionales *#ev3*	2021-05-11
27. Programación de interfaces gráficas de usuario *#ev3*	2021-05-18
28. Entrada y salida de información *#ev1*	2020-11-03
29. Complejidad algorítmica *#opcional*	
30. Metodología de la programación *#opcional*	
31. Principios y patrones de diseño *#opcional*	

5.2. Esquema detalle

El título de cada unidad va etiquetado con los resultados de aprendizaje y los criterios de evaluación relacionados con dicha unidad. Cuando un resultado de aprendizaje no lleva asociado ningún criterio de evaluación, significa que dicho resultado de aprendizaje se trabaja en la unidad didáctica pero no es el elemento fundamental de evaluación.

1 **Introducción** *#ev1 #ra1* (est: 2020–09–21)

1.1. Conceptos básicos

1.1.1. Informática

- Procesamiento automático

1.1.2. Ordenador

- Definición
- Funcionamiento básico
  - Elementos funcionales
  - Ciclo de instrucción
  - Representación de información
    - Codificación interna
      - Sistema binario
    - Codificación externa
      - ASCII
      - Unicode

1.1.3. Problema

- Generalización
- Ejemplares de un problema
- Dominio de definición
- Jerarquías de generalización

1.1.4. Algoritmo

- Definición
- Características
- Representación
  - Ordinograma
  - Pseudocódigo
- Cualidades deseables
- Computabilidad
- Corrección
- Complejidad

1.1.5. Programa
1.1.6. Lenguaje de programación

1.2. Paradigmas de programación
1.2.1. Definición
1.2.2. Imperativo
- Estructurado
- Orientado a objetos

1.2.3. Declarativo
- Funcional
- Lógico
- De bases de datos

1.3. Lenguajes de programación
1.3.1. Definición
- Sintaxis
  - Notación EBNF
- Semántica estática
- Semántica dinámica

1.3.2. Evolución histórica
1.3.3. Clasificación
- Por nivel
- Por generación
- Por propósito
- Por paradigma

1.4. Traductores
1.4.1. Definición
1.4.2. Compiladores
- Ensambladores

1.4.3. Intérpretes
- Interactivos (*REPL*)

1.5. Resolución de problemas mediante programación
1.5.1. Especificación
1.5.2. Análisis del problema
1.5.3. Diseño del algoritmo
1.5.4. Verificación
1.5.5. Estudio de la eficiencia
1.5.6. Codificación

1.5.7. Traducción y ejecución
1.5.8. Pruebas
1.5.9. Depuración
1.5.10. Documentación
1.5.11. Mantenimiento
1.5.12. Ingeniería del software

1.6. Entornos integrados de desarrollo
1.6.1. Definición
1.6.2. Editores de textos
1.6.3. Editores vs. IDE
1.6.4. Visual Studio Code

2 **Expresiones** *#ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2020–09–28)

2.1. El lenguaje de programación Python
2.1.1. Historia
2.1.2. Características principales
2.1.3. Instalación
2.1.4. Funcionamiento del intérprete
- Entrada y salida del intérprete

2.1.5. Instalación de Visual Studio Code
- Configuración básica de Visual Studio Code

2.2. Elementos de un programa
2.2.1. Expresiones y sentencias
2.2.2. Sintaxis y semántica de las expresiones

2.3. Valores
2.3.1. Datos, tipos y valores
2.3.2. Evaluación de expresiones
2.3.3. Expresión canónica y forma normal
2.3.4. Formas normales y evaluación
2.3.5. Literales
2.3.6. Identificadores

2.4. Operaciones
2.4.1. Clasificación
2.4.2. Operadores
- Aridad de operadores
- Notación de los operadores

- Paréntesis
- Prioridad de operadores
- Asociatividad de operadores
- Paréntesis y orden de evaluación
- Tipos de operandos

2.4.3. Funciones

- Funciones con varios parámetros
- Evaluación de expresiones con funciones
- Composición de operaciones y funciones

2.4.4. Métodos

2.5. Otros conceptos sobre operaciones

2.5.1. Sobrecarga de operaciones

2.5.2. Equivalencia entre formas de operaciones

2.5.3. Igualdad de operaciones

2.6. Operaciones predefinidas

2.6.1. Operadores predefinidos

- Operadores aritméticos
- Operadores de cadenas

2.6.2. Funciones predefinidas

- Funciones matemáticas y módulos
  - El módulo `operator`

2.6.3. Métodos predefinidos

3 Programación funcional (I) *#ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2020–10–05)

3.1. Introducción

3.1.1. Concepto

3.1.2. Transparencia referencial

3.1.3. Modelo de ejecución

- Modelo de sustitución

3.2. Tipos de datos

3.2.1. Concepto

3.2.2. `type`

3.2.3. Sistemas de tipos

- Errores de tipos
- Tipado fuerte vs. débil

3.2.4. Tipos de datos básicos

- Números
- Cadenas
- Funciones

3.2.5. Conversión de tipos

3.3. Álgebra de Boole

3.3.1. El tipo de dato *booleano*

3.3.2. Operadores relacionales

3.3.3. Operadores lógicos

- Tablas de verdad

3.3.4. Axiomas

- Traducción a Python

3.3.5. Teoremas fundamentales

- Traducción a Python

3.3.6. El operador ternario

3.4. Definiciones

3.4.1. Introducción

3.4.2. Identificadores y ligaduras (*binding*)

- Reglas léxicas
- Tipo de un identificador
- Las funciones como datos

3.4.3. Evaluación de expresiones con identificadores

3.4.4. Marcos (*frames*)

3.4.5. *Scripts*

3.5. Documentación interna

3.5.1. Identificadores significativos

3.5.2. Comentarios

4 **Programación funcional (II)** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2020–10–12)

4.1. Abstracciones lambda

4.1.1. Expresiones lambda

4.1.2. Parámetros y cuerpos

4.1.3. Aplicación funcional

- Evaluación de una aplicación funcional
- Llamadas a funciones

4.1.4. Variables ligadas y libres

4.2. Ámbitos léxicos

4.2.1. Ámbitos

4.2.2. Ámbito de una ligadura y de creación de una ligadura

4.2.3. Ámbito de un identificador

4.2.4. Ámbito de un parámetro

4.2.5. Ámbito de una variable ligada

4.2.6. Entorno (*environment*)

4.2.7. Ámbitos, marcos y entornos

4.2.8. Variables *sombreadas*

4.2.9. Renombrado de parámetros

4.3. Evaluación

4.3.1. Evaluación de expresiones con entornos

4.3.2. Evaluación de expresiones lambda con entornos

- Visualización en *Pythontutor*

4.3.3. Estrategias de evaluación

- Orden de evaluación
  - Orden aplicativo
  - Orden normal
- Composición de funciones
- Evaluación estricta y no estricta

4.4. Abstracciones funcionales

4.4.1. Pureza

4.4.2. Las funciones como abstracciones

- Especificaciones de funciones

4.5. Computabilidad

4.5.1. Funciones y procesos

4.5.2. Funciones *ad-hoc*

4.5.3. Funciones recursivas

- Definición
- Casos base y casos recursivos
- El factorial
- Diseño de funciones recursivas
  - Pensamiento optimista
  - Descomposición del problema
  - Identificación de problemas no reducibles

- Recursividad lineal
  - Procesos recursivos lineales
  - Procesos iterativos linales
- Recursividad múltiple
- Recursividad final y no final

4.5.4. La pila de control

4.5.5. Un lenguaje Turing-completo

4.6. Tipos de datos recursivos

4.6.1. Concepto

4.6.2. Cadenas

4.6.3. Listas

4.6.4. Rangos

4.6.5. Conversión a lista

4.7. Funciones de orden superior

4.7.1. Concepto

4.7.2. `map`

4.7.3. `filter`

4.7.4. `reduce`

4.7.5. Expresiones generadoras

5 **Programación imperativa** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5a #ce5b #ce5c #ce5d #ce5e #ce6h #ce6i #ev1 #ra1 #ra3 #ra5 #ra6* (est: 2020–10–19)

5.1. Modelo de ejecución

5.1.1. Máquina de estados

5.1.2. Sentencias

5.1.3. Secuencia de sentencias

5.2. Asignación destructiva

5.2.1. Identidad

- `id`

5.2.2. Variables y referencias

5.2.3. Estado

5.2.4. Marcos en programación imperativa

5.2.5. Sentencia de asignación

5.2.6. Evaluación de expresiones con variables

5.2.7. Constantes

5.2.8. Tipado estático vs. dinámico

5.2.9. Asignación compuesta

5.2.10. Asignación múltiple

5.3. Mutabilidad

5.3.1. Estado de un dato

5.3.2. Tipos mutables e inmutables

- Inmutables
- Mutables

5.3.3. Alias de variables

- Recolección de basura
- `is`

5.4. Cambios de estado ocultos

5.4.1. Funciones puras

5.4.2. Funciones impuras

5.4.3. Efectos laterales

5.4.4. Transparencia referencial

5.4.5. Entrada y salida por consola

- `print`
  - Paso de argumentos por palabras clave
  - El valor `None`
- `input`

5.4.6. Ejecución de *scripts* por lotes

- Argumentos de la línea de órdenes

5.4.7. Entrada y salida por archivos

- `open`
- `read`
- `readline`
- `write`
- `seek` y `tell`
- `close`

5.5. Saltos

5.5.1. Incondicionales

5.5.2. Condicionales

6 **Programación estructurada** *#ce1a #ce1b #ce1c #ce3a #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2020–10–26)

6.1. Aspectos teóricos de la programación estructurada

6.1.1. Programación estructurada

- De aceptación

9.3.3. `doctest`

9.3.4. `pytest`

9.3.5. Desarrollo conducido por pruebas

- Ciclo de desarrollo
- Ventajas

10 **Abstracción de datos** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2020–11–23)

10.1. Introducción

10.1.1. Introducción

10.1.2. Tipos abstractos de datos

10.2. Especificaciones

10.2.1. Sintaxis

10.2.2. Operaciones

10.2.3. Ejemplos

10.3. Implementaciones

10.3.1. Implementaciones

10.4. Niveles y barreras de abstracción

10.4.1. Niveles de abstracción

10.4.2. Barreras de abstracción

10.4.3. Propiedades de los datos

10.5. Las funciones como datos

10.5.1. Clausuras

10.5.2. Representación funcional

10.5.3. Estado interno

10.5.4. Paso de mensajes

10.5.5. Especificación de datos abstractos con estado interno

10.6. Abstracción de datos y modularidad

10.6.1. El tipo abstracto como módulo

11 **Programación orientada a objetos** *#ce1a #ce1b #ce1c #ce2a #ce2b #ce2c #ce2d #ce2f #ce2h #ce2i #ce3f #ce3g #ce6a #ev2 #ra1 #ra2 #ra3 #ra6* (est: 2021–01–11)

11.1. Introducción

11.1.1. Recapitulación

11.1.2. La metáfora del objeto

11.2. Clases y objetos

11.2.1. Clases

13.2.5. El entorno de ejecución de Java (JRE)

- El intérprete `java`

13.3. El primer programa Java

13.3.1. Tipado estático vs. dinámico

13.3.2. El método `main`

13.3.3. La clase principal

13.3.4. La clase `System`

13.3.5. El paquete `java.lang`

13.3.6. El objeto `out`

13.3.7. El método `println`

14 Elementos básicos del lenguaje Java *#ce1a #ce1b #ce1c #ce1d #ce1e #ce1f #ce1h #ce1i #ce2b #ce2i #ce3a #ce3b #ce3c #ce3e #ce3f #ce3g #ev2 #ra1 #ra2 #ra3 #ra5* (est: 2021–02–01)

14.1. Tipos y valores en Java

14.1.1. Introducción

14.1.2. Tipos primitivos

- Booleanos
- Integrales
  - Operadores de integrales
- De coma flotante
  - Operadores de coma flotante
- Subtipado
  - Subtipado entre tipos primitivos
  - Subtipado entre tipos referencia
- Conversiones entre datos primitivos
  - *Casting*
  - De ampliación (*widening*)
  - De restricción (*narrowing*)
- Promociones numéricas

14.1.3. Tipos referencia

- Nulo
- Acceso a miembros
  - Llamadas a métodos sobrecargados

14.2. Variables en Java

14.2.1. Introducción

14.2.2. Variables de tipos primitivos

14.2.3. Variables de tipos referencia
- Tipo estático y tipo dinámico

14.2.4. Declaración de variables
- Inicialización y asignación de variables
  - Operadores de asignación compuesta
  - Operadores de incremento y decremento
  - Inicialización y asignación con literales numéricos
- Inferencia de tipos
- Constantes
- Declaración de variables de tipo referencia

14.3. Estructuras de control

14.3.1. Bloques

14.3.2. `if`

14.3.3. `switch`

14.3.4. `while`

14.3.5. `for`

14.3.6. `do ... while`

14.3.7. `break` y `continue`

14.4. Entrada/salida

14.4.1. Flujos `System.in`, `System.out` y `System.err`

14.4.2. `java.util.Scanner`

15 Programación orientada a objetos en Java *#ce1a #ce1b #ce1c #ce2e #ce3f #ce3g #ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h #ev2 #ra1 #ra2 #ra3 #ra4* (est: 2021–02–08)

15.1. Uso básico de objetos

15.1.1. Instanciación
- `new`
- `getClass`
- `instanceof`

15.1.2. Referencias
- `null`

15.1.3. Comparación de objetos
- `equals`
- `hashCode`

15.1.4. Destrucción de objetos y recolección de basura

15.2. Clases y objetos básicos en Java

15.2.1. Cadenas

- Inmutables
- Mutables
  - `StringBuffer`
  - `StringBuilder`
  - `StringTokenizer`
- Conversión a `String`
- Concatenación de cadenas
- Comparación de cadenas
- Diferencias entre literales cadena y objetos `String`

15.2.2. Clases envolventes (*wrapper*)

- *Boxing* y *unboxing*
- *Autoboxing* y *autounboxing*
- La clase `Number`

15.3. *Arrays*

15.3.1. Definición

15.3.2. Declaración

15.3.3. Creación

15.3.4. Inicialización

15.3.5. Acceso a elementos

15.3.6. Longitud de un _array_

15.3.7. Modificación de elementos

15.3.8. _Arrays_ de tipos referencia

15.3.9. Subtipado entre _arrays_

15.3.10. `java.util.Arrays`

15.3.11. Copia y redimensionado de _arrays_

- `clone`
- `System.arraycopy`)
- `Arrays.copyOf`

15.3.12. Comparación de *arrays*

- `Arrays.equals`

15.3.13. _Arrays_ multidimensionales

- Declaración
- Creación
- Inicialización
- `Arrays.deepEquals`

16 **Diseño de clases en Java** (est: 2021–02–15)

16.1. Definición de clases

16.1.1. Sintaxis básica

16.1.2. Visibilidad de una clase

- Visibilidad predeterminada
- Visibilidad pública

16.1.3. Visibilidad de un miembro de una clase

16.2. Miembros de instancia

16.2.1. Variables de instancia

- Acceso y modificación
- Variables de instancia finales

16.2.2. Métodos de instancia

- Invocación
- La sentencia `return`
- Referencia `this`
- Ámbito y resolución de identificadores
- Accesores y mutadores
- Sobrecarga
- Constructores
  - Sobrecarga de constructores
  - Constructor por defecto

16.3. Miembros estáticos

16.3.1. Métodos estáticos

16.3.2. Variables estáticas

16.3.3. Variables estáticas finales

16.4. Clases internas

16.4.1. Clases internas anidadas

16.4.2. Clases anidadas estáticas

17 **Relaciones entre clases en Java** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce4g #ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h #ev2 #ra1 #ra3 #ra4 #ra7* (est: 2021–02–22)

17.1. Asociaciones básicas

17.1.1. Agregación

17.1.2. Composición

17.2. Herencia

17.2.1. Subtipado entre tipos referencia

19 **Programación genérica** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6f #ev2 #ra1 #ra3 #ra6* (est: 2021–03–09)

19.1. Tipos genéricos

19.1.1. Parámetros de tipo

19.1.2. Argumentos de tipo

19.1.3. Tipos crudos

19.2. Métodos genéricos

19.3. Subtipos

19.3.1. Parámetros de tipo acotados

19.3.2. Clases genéricas, herencia y subtipos

- Covarianza
- Contravarianza
- Invarianza

19.4. Inferencia de tipos

19.5. Comodines

19.6. Borrado de tipos

19.7. Limitaciones

20 **Estructuras de datos lineales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ev2 #ra1 #ra3 #ra6* (est: 2021–03–16)

20.1. Acceso secuencial

20.1.1. Listas

- Enlazadas
- Doblemente enlazadas

20.1.2. Pilas

20.1.3. Colas

20.2. Acceso directo

20.2.1. Tablas *hash*

21 **Ordenación y búsqueda** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3 #ra6* (est: 2021–04–05)

21.1. Algoritmos de búsqueda

21.1.1. Búsqueda secuencial

21.1.2. Búsqueda dicotómica

21.2. Algoritmos de ordenación

21.2.1. Inserción directa

21.2.2. Selección directa

21.2.3. Burbuja
21.2.4. *Quicksort*
21.2.5. *Mergesort*
21.3. Tablas *Hash*

22 **Estructuras de datos no lineales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3 #ra6*
(est: 2021–04–12)

22.1. Árboles
22.1.1. Binarios
- Recorridos
  - Preorden
  - Inorden
  - Postorden

22.1.2. De búsqueda
22.1.3. Montículos
- Algoritmo de ordenación

22.1.4. Generales
- Recorrido en profundidad
- Recorrido en anchura

22.2. Grafos
22.2.1. Algoritmo de Dijkstra
22.2.2. Algoritmo de Floyd

23 **Control de excepciones en Java** *#ce1a #ce1b #ce1c #ce3d #ce3f #ce3g #ev3 #ra1 #ra3*
(est: 2021–04–19)

23.1. Errores y excepciones
23.2. El requisito «*captura o especifica*»
23.2.1. Tipos de excepciones
23.3. Captura y manejo de excepciones
23.3.1. Bloque `try`
23.3.2. Bloques `catch`
23.3.3. Bloque `finally`
23.4. Excepciones y signaturas
23.5. Lanzamiento de excepciones
23.5.1. Excepciones encadenadas
23.5.2. Creación de clases de excepción

23.6. Excepciones no chequeadas

23.7. Ventajas de las excepciones

24 **Java Collections Framework** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ce6b #ce6c #ce6d #ce6e #ce6f #ev3 #ra1 #ra3 #ra6* (est: 2021–04–26)

24.1. Colecciones y *arrays*

24.2. Arquitectura

24.3. Tipos de colecciones

24.3.1. Listas ordenadas

24.3.2. Conjuntos

24.3.3. Diccionarios

24.4. Listas

24.4.1. `java.util.List`

- `java.util.ArrayList`
- `java.util.LinkedList`
- `java.util.Stack`

24.5. Colas

24.5.1. Interfaz `java.util.Queue`

- `java.util.ArrayDeque`
- `java.util.PriorityQueue`
- `java.util.LinkedList`

24.5.2. Interfaz `java.util.Deque`

- `java.util.ArrayDeque`
- `java.util.LinkedList`

24.6. Conjuntos

24.6.1. Interfaz `java.util.Set`

- `java.util.HashSet`
- `java.util.LinkedHashSet`
- `java.util.TreeSet`

24.6.2. Interfaz `java.util.SortedSet`

- `java.util.TreeSet`

24.6.3. Interfaz `java.util.NavigableSet`

- `java.util.TreeSet`

24.7. Diccionarios

24.7.1. Interfaz `java.util.Map`

- `java.util.HashMap`

- `java.util.LinkedHashMap`
- `java.util.TreeMap`

24.7.2. Interfaz `java.util.SortedMap`

- `java.util.TreeMap`

24.7.3. Interfaz `java.util.NavigableMap`

- `java.util.TreeMap`

25 **Calidad (II)** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3* (est: 2021–05–04)

25.1. Pruebas automáticas

25.1.1. JUnit

25.2. Documentación

25.2.1. Interna

- Reglas de estilo
- Google Java Format

25.2.2. Externa

- Javadoc

26 **Gestión de bases de datos relacionales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce8a #ce8b #ce8c #ce8d #ce8e #ce8f #ce8g #ce8h #ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g #ev3 #ra1 #ra3 #ra6 #ra8 #ra9* (est: 2021–05–11)

26.1. Controlador JDBC

26.1.1. Instalación

26.1.2. `CLASSPATH`

26.1.3. Carga

26.2. Establecimiento de conexiones

26.3. Recuperación de información

26.3.1. Ejecución de consultas

26.3.2. Selección de registros

26.3.3. Uso de parámetros

26.4. Manipulación de la información

26.4.1. Altas, bajas y modificaciones

27 **Programación de interfaces gráficas de usuario** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5f #ce5g #ce5h #ev3 #ra1 #ra3 #ra5 #ra6* (est: 2021–05–18)

27.1. JFC y Swing

27.2. Componentes de Swing

29.7. Resolución de recurrencias

29.7.1. Reducción de problemas mediante sustracción

29.7.2. Reducción de problemas mediante división

30 Metodología de la programación *#ce1a #ce1b #ce1c #ce3f #ce3g #opcional #ra1 #ra3 #ra6*

30.1. Ciclo de vida

30.2. Especificación e implementación

30.3. Verificación y validación de programas

30.3.1. Demostraciones por inducción

30.4. Programación funcional

30.4.1. Especificaciones formales

- Como cálculo

30.4.2. Derivación de programas

- Diseño recursivo
  - Recursividad final
  - Técnicas de inmersión *#opcional*

30.5. Programación imperativa

30.5.1. Especificaciones formales

- Como modificación de estados

30.5.2. Derivación de programas

- Diseño iterativo
  - Invariante de un bucle
  - Transformación de recursividad final a iterativo

30.6. El lenguaje Dafny *#opcional*

31 Principios y patrones de diseño *#ce1a #ce1b #ce1c #ce3f #ce3g #opcional #ra1 #ra3*

31.1. Principios de diseño

31.1.1. Encapsulación y ocultación de información

31.1.2. Diseño orientado a interfaces

31.1.3. Principios *SOLID*

- SRP: Principio de responsabilidad única
- OCP: Principio de abierto/cerrado
- LSP: Principio de sustitución de Liskov
- ISP: Principio de segregación de la interfaz
- DIP: Principio de inversión de dependencias

31.1.4. Principio del Menor Conocimiento (o Ley de Demeter)

31.2. Patrones de diseño

31.2.1. De creación

31.2.2. Estructurales

31.2.3. De comportamiento

5.3. Correspondencia con resultados de aprendizaje y criterios de evaluación

El símbolo «×» representa que en esa unidad didáctica se trabaja dicho resultado de aprendizaje pero no es el elemento fundamental de evaluación.

Unidades didácticas	*#ra1*	*#ra2*	*#ra3*	*#ra4*	*#ra5*	*#ra6*	*#ra7*	*#ra8*	*#ra9*
1. Introducción	×								
2. Expresiones	*#ce1a* *#ce1b* *#ce1c* *#ce1e* *#ce1f* *#ce1g* *#ce1i*		*#ce3f* *#ce3g*			×			
3. Programación funcional (I)	*#ce1a* *#ce1b* *#ce1c* *#ce1e* *#ce1f* *#ce1g* *#ce1i*		*#ce3f* *#ce3g*			×			
4. Programación funcional (II)	*#ce1a* *#ce1b* *#ce1c*		*#ce3f* *#ce3g*			×			
5. Programación imperativa	*#ce1a* *#ce1b* *#ce1c*		*#ce3f* *#ce3g*		*#ce5a* *#ce5b* *#ce5c* *#ce5d* *#ce5e*	*#ce6h* *#ce6i*			
6. Programación estructurada	*#ce1a* *#ce1b* *#ce1c*		*#ce3a* *#ce3f* *#ce3g*			×			
7. Tipos de datos estructurados	*#ce1d* *#ce1h*		*#ce3f* *#ce3g*			*#ce6g*			
8. Programación modular (I)	*#ce1a* *#ce1b* *#ce1c*		*#ce3f* *#ce3g*			×			

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
9. Calidad (I)	#ce1a #ce1b #ce1c		#ce3f #ce3g						
10. Abstracción de datos	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
11. Programación orientada a objetos	#ce1a #ce1b #ce1c	#ce2a #ce2b #ce2c #ce2d #ce2f #ce2h #ce2i	#ce3f #ce3g			#ce6a			
12. Relaciones entre clases	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4g			#ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h		
13. Introducción a la tecnología Java	#ce1a #ce1b #ce1c #ce1e #ce1f	#ce2b #ce2i							
14. Elementos básicos del lenguaje Java	#ce1a #ce1b #ce1c #ce1d #ce1e #ce1f #ce1h #ce1i	#ce2b #ce2i	#ce3a #ce3b #ce3c #ce3e #ce3f #ce3g		×				
15. Programación orientada a objetos en Java	#ce1a #ce1b #ce1c	#ce2e	#ce3f #ce3g	#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h					
16. Diseño de clases en Java									

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
17. Relaciones entre clases en Java	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4g			#ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h		
18. Programación modular (II)	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4i #ce4j					
19. Programación genérica	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6f			
20. Estructuras de datos lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a			
21. Ordenación y búsqueda	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
22. Estructuras de datos no lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
23. Control de excepciones en Java	#ce1a #ce1b #ce1c		#ce3d #ce3f #ce3g						
24. Java Collections Framework	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a #ce6b #ce6c #ce6d #ce6e #ce6f			
25. Calidad (II)	#ce1a #ce1b #ce1c		#ce3f #ce3g						
26. Gestión de bases de datos relacionales	#ce1a #ce1b #ce1c		#ce3f #ce3g			×		#ce8a #ce8b #ce8c #ce8d #ce8e #ce8f #ce8g #ce8h	#ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
27. Programación de interfaces gráficas de usuario	#ce1a #ce1b #ce1c		#ce3f #ce3g		#ce5f #ce5g #ce5h	×			
28. Entrada y salida de información	#ce1a #ce1b #ce1c		#ce3f #ce3g		#ce5a #ce5b #ce5c #ce5d #ce5e	#ce6h #ce6i			
29. Complejidad algorítmica									
30. Metodología de la programación	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
31. Principios y patrones de diseño	#ce1a #ce1b #ce1c		#ce3f #ce3g						

6. Orientaciones pedagógicas

Según queda recogido en [1], este módulo profesional contiene parte de la formación necesaria para desempeñar la función de **programación de aplicaciones de propósito general en lenguajes orientados a objetos**.

La **función** de programación de aplicaciones de propósito general en lenguajes orientados a objetos incluye aspectos como:

- El desarrollo de programas organizados en clases aplicando los principios de la programación orientada a objetos.
- La utilización de interfaces para la interacción de la aplicación con el usuario.
- La identificación, análisis e integración de librerías para incorporar funcionalidades específicas a los programas desarrollados.
- El almacenamiento y recuperación de información en sistemas gestores de bases de datos relacionales y orientados a objetos.

Las **actividades profesionales** asociadas a esta función se aplican en el desarrollo y la adaptación de programas informáticos de propósito general en lenguajes orientados a objetos.

Las **líneas de actuación** en el proceso de enseñanza-aprendizaje que permiten alcanzar los objetivos del módulo profesional versarán sobre:

- La interpretación y aplicación de los principios de la programación orientada a objetos.

- La evaluación, selección y utilización de herramientas y lenguajes de programación orientados a objetos.
- La utilización de las características específicas de lenguajes y entornos de programación en el desarrollo de aplicaciones informáticas.
- La identificación de las funcionalidades aportadas por los sistemas gestores de bases de datos y su incorporación a los programas desarrollados.
- La documentación de los programas desarrollados.

7. Orientaciones metodológicas

El módulo es totalmente práctico, y el proceso de enseñanza-aprendizaje se fundamenta en la interacción continua y total de los alumnos con las herramientas software utilizadas y estudiadas a lo largo del curso.

El módulo construye el aprendizaje de forma progresiva, comenzando con el estudio de los fundamentos de la programación en un núcleo funcional sencillo para, desde ahí, ir incorporando nuevos elementos paulatinamente aumentando poco a poco la complejidad, añadiendo primero los elementos básicos de la programación imperativa, introduciendo luego las ventajas de la programación estructurada y, finalmente, incorporando los mecanismos de la programación orientada a objetos. La idea, por tanto, es hacer que cada nueva capa de complejidad se apoye en la anterior.

De la misma forma, el estudio de los datos se hará añadiendo complejidad progresiva, partiendo de los tipos de datos más basicos, introduciendo luego los tipos compuestos, estudiando los tipos abstractos de datos y acabando con la implementación de las estructuras de datos clásicas. Es en ese punto donde este enfoque se encuentra con el del párrafo anterior, al hacer notar que las clases del paradigma de programación orientada a objetos son, en esencia, implementaciones de tipos abstractos de datos.

Para servir de vehículo de comunicación y soporte de los conceptos aprendidos durante el curso se utilizarán dos lenguajes de programación: Python durante el primer trimestre y Java en los dos restantes. Los motivos que justifican el uso de Python como lenguaje de introducción a la programación son los siguientes:

- Es un lenguaje sencillo y fácil de aprender en cuanto a sus aspectos básicos.
- Es el lenguaje más usado por universidades y centros de formación en cursos de introducción a la programación.
- Es un lenguaje interpretado y dispone de intérprete interactivo, lo que facilita el desarrollo iterativo e incremental.

- Es un lenguaje de tipado dinámico pero dispone de mecanismos opcionales de tipado estático, por lo que pasar de uno a otro resulta sencillo.
- Es un lenguaje multiparadigma, lo que permite transitar por los estilos funcional, imperativo, estructurado y orientado a objetos.
- Es, a día de hoy, el lenguaje con mayor crecimiento en la industria del software, superando a Java y JavaScript.
- Es muy usado en la industria y dispone de muchas y probadas herramientas de desarrollo, así como librerías y paquetes de muy variada funcionalidad.
- Permite un acceso muy fácil a bases de datos relacionales.

No obstante lo anterior, se reconoce la importancia del lenguaje de programación Java en la industria del desarrollo de software orientado a objetos, por lo que resulta especialmente interesante usar dicho lenguaje en el estudio de ese paradigma de programación. Debido a ello, los trimestres segundo y tercero usarán Java, partiendo de todo lo aprendido en unidades anteriores con el lenguaje Python.

De manera transversal, se hará hincapié continuamente en el aseguramento de la calidad del producto resultante así como del proceso de desarrollo y el código fuente desarrollado, poniendo especial énfasis en la metodología *TDD (Test-Driven Development)* y las pruebas automáticas.

La enseñanza se basará casi por completo en el estudio previo de los conceptos básicos y suficientes para la realización de ejercicios y supuestos de aplicación, que obliguen al alumno a enfrentarse con las herramientas software necesarias para solucionarlos.

Las actividades propuestas podrán ser individuales o grupales, aplicando donde corresponda el concepto de *programación en pareja*[1] como vehículo de colaboración y aprendizaje compartido entre varios alumnos.

Finalmente, se incentivará al alumno para que mejore su comprensión mediante el autoaprendizaje y la elaboración propia de ejercicios y desarrollos.

8. Recursos

8.1. Hardware

- Un ordenador para cada alumno, conectado a la red local del aula y esta, a su vez, a la troncal del Centro.
- Conexión a Internet de banda ancha.
- Cañón retroproyector.

[1] https://es.wikipedia.org/wiki/Programaci%C3%B3n_en_pareja

8.2. Software

- Sistema operativo GNU/Linux (Ubuntu o Debian GNU/Linux, preferentemente).
- El resto de herramientas y aplicaciones necesarias se instalarán a través de Internet a lo largo del curso.

8.3. Online

- Plataforma **Moodle Centros**[2] de la Junta de Andalucía para el seguimiento general del módulo, incluyendo distribución de material y entrega de ejercicios y exámenes.
- **GitHub** y **GitHub Classroom**[3] como herramientas centralizadas para compartir código y para la gestión integral de todo elemento satélite del mismo (control de versiones, desarrollo colaborativo, incidencias, etcétera).

8.4. Bibliografía

8.4.1. Principal

- Apuntes y documentación *online*[4] suministrados por el profesor.
- Documentación oficial de Python[5].
- Documentación oficial de Java SE[6].

8.4.2. Complementaria

- Abelson, H., Sussman, G. J. y Sussman, J. (1996). *Structure and Interpretation of Computer Programs (2nd edition)*. Cambridge: MIT Press.
- Peña, R. (2003). *Diseño de programas: formalismo y abstracción (2.ª edición)*. Madrid: Prentice-Hall.
- Blanco, J., Smith, S., Barsotti, D. (2009). *Cálculo de Programas*. Córdoba (Argentina): Fa.M.A.F., Universidad Nacional de Córdoba.
- Pareja, C., Ojeda, M., Andeyro, Á. L., Rossi, C. (1997). *Desarrollo de algoritmos y técnicas de Programación en Pascal*. Madrid: Ra-Ma.

[2] https://educacionadistancia.juntadeandalucia.es/centros/cadiz
[3] https://classroom.github.com
[4] https://pro.iesdonana.org
[5] https://docs.python.org/3/
[6] https://docs.oracle.com/en/java/javase/

- Joyanes Aguilar, L. (2008). *Fundamentos de programación. Algoritmos, estructuras de datos y objetos*. Aravaca: McGraw-Hill Interamericana de España.
- Van-Roy, P., Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. Cambridge, Mass: MIT Press.

9. Atención a la diversidad

Se llevarán a cabo actividades de refuerzo o ampliación para aquellos alumnos que así lo requieran en función de las necesidades detectadas:

- Para los alumnos que muestren dificultades de aprendizaje, se propondrán **actividades de refuerzo** destinadas a afianzar los aspectos conceptuales y procedimentales en los que el alumnado presente carencias.
- Para los alumnos que muestren un mayor grado de adquisición de competencias, se propondrán **actividades de ampliación** que supongan la investigación autónoma de contenidos opcionales (aquellos marcados con la etiqueta *#opcional*).

10. Temas transversales

Con el objeto de fomentar entre los alumnos el hábito de la lectura, se plantearán actividades individuales y en grupo en las que, para su resolución, se necesite leer información de distintas fuentes escritas, como artículos, blogs, páginas web, tutoriales, etc.

La evolución experimentada por la informática en los últimos años tiene como consecuencia su influencia inevitable en todos los aspectos de las relaciones entre las personas y entre éstas y el entorno. Además ha demostrado ser un medio valiosísimo para la educación cualquiera que sea el ámbito en el que se use. En concreto, en cuanto a los temas transversales propuestos:

- **Educación ambiental:** La utilización de la informática, en general, y sobre todo en los negocios, hace que grandes volúmenes de información puedan ser almacenados en soportes informáticos, discos, CD, … y enviados de unos lugares a otros a través de las redes informáticas, evitándose de esta manera el consumo de grandes cantidades de papel y, por consiguiente, la destrucción de bosques, contribuyendo de alguna manera a la preservación de los medios naturales y medioambientales.
- **Educación del consumidor:** El análisis y la utilización de diferentes herramientas informáticas favorecen la capacidad del alumnado para decidir sobre los productos informáticos que debe adquirir y utilizar de manera ventajosa.

- **Educación para la salud:** Cuando se utilizan equipos informáticos se procura que el alumnado conozcan una serie de normas de higiene y seguridad en el trabajo, así como sobre las precauciones necesarias en el empleo de los equipos. De esta manera, se intenta que el alumnado conozca los principios de la ergonomía del puesto de trabajo, para que cualquier trabajo frente al ordenador resulte lo más agradable posible y no le cause ningún problema. En este sentido, resultan de interés las instrucciones elaboradas por el Instituto Nacional de Seguridad e Higiene en el Trabajo [3].
- **Educación para la igualdad de oportunidades entre ambos sexos:** Desde este módulo contamos con elementos para concienciar al alumnado sobre la igualdad de oportunidades para alumnos y alumnas:
  - Formando grupos mixtos de trabajo.
  - Distribuyendo las tareas a realizar en la misma medida entre el alumnado de ambos sexos.
  - Haciendo que todos utilicen los mismos o equivalentes equipos.
  - Fomentando la participación de todos, sin distinciones de sexo.
- **Educación para el trabajo:** Respecto a este módulo encontramos los siguientes elementos:
  - Técnicas de trabajo en grupo: sujeción a unas reglas corporativas.
  - Colaboración de varias personas para la realización de un único trabajo.
- **Educación para la paz y la convivencia:** Se trabajan los elementos siguientes:
  - Acuerdos para la utilización de los mismos estándares en toda la comunidad internacional.
  - Respeto por las opiniones de los demás.
  - Aprender a escuchar.

11. Actuaciones para desarrollar la perspectiva de género

El conocimiento de la realidad existente es el primer paso a realizar para incorporar la perspectiva de género. De esta manera, se descubrirá la existencia de situaciones de desequilibrio entre mujeres y hombres en el desempeño de la actividad docente.

La perspectiva de género es trabajada de manera transversal y permanente en todas las Unidades Didácticas que componen esta programación. El IES Doñana como organización social en aplicación de esta óptica favorece, entre otros aspectos, la detección de estereotipos y la asignación de roles y responsabilidades, la evaluación del uso y control de los recursos puestos a

disposición de hombres y mujeres con la finalidad última de introducir las modificaciones y medidas correctoras necesarias para eliminar las desigualdades detectadas en cualquier ámbito de la vida del centro y particularmente dentro del aula.

En el marco de esta programación, el análisis de estas circunstancias permite identificar las diferentes necesidades, intereses y perspectivas de mujeres y hombres sobre las que diseñar estrategias que equiparen las oportunidades de ambas partes en las distintas actuaciones que lo integran. Fundamentalmente en los siguientes círculos se realizan las actuaciones:

- Profesores–profesores
- Alumnos–alumnos y
- Alumnos–profesores

Implica tener en cuenta las siguientes cuestiones:

1. Valorar la situación de partida de hombres y mujeres.
2. Analizar las necesidades y obligaciones relacionadas con la actividad cotidiana en el centro y la posición social de hombres y mujeres en el centro.
3. Velar por el cumplimiento de la condición de igualdad de género en todos los ámbitos de actuación como cuestión de justicia y responsabilidad social.

11.1. Actuaciones generales permanentes

1. Revisión del material curricular para la eliminación de la transmisión de estereotipos o modelos de conductas determinados por el género, tipo identificación cultural de funciones realizadas tradicionalmente por hombres o mujeres.
2. Detectar las desigualdades y discriminaciones de género existentes en el centro para su tratamiento/denuncia pertinente.
3. Garantizar la participación equilibrada de hombres y mujeres en las distintas actividades en el aula y en el centro.
4. Velar porque el contenido gráfico y lingüístico de las acciones, materiales y dispositivos de formación y difusión carezca de cualquier carácter o pretensión discriminatoria.
5. Participación en las actividades propuestas por el Plan de Igualdad del centro articulado a través de actuaciones propias o la acción tutorial:
   a) 25 de noviembre: Violencia de Género.
   b) 30 de enero: Resolución de conflictos de forma pacífica. Día de la Paz.
   c) 8 de marzo: Día de la Mujer Trabajadora.

Desde el primer momento se advertirá al alumnado que el uso del vocabulario y expresiones propias del lenguaje hablado y escrito se llevará a cabo de forma extensiva a ambos géneros, de manera que cuando hablamos del «administrador» o el «programador» lo hacemos siempre considerando que dichos roles son de aplicación a hombres y mujeres por igual. Así pues, resultará innecesario y, por tanto, se evitará el uso de fórmulas tales como «administrador o administradora», que recargan el lenguaje sin aportar información adicional. Ello además va en consonancia con lo manifestado por la Real Academia Española, al afirmar que:

> «*El español dispone de un mecanismo inclusivo: el masculino gramatical, que, como término no marcado de la oposición de género, puede referirse a grupos formados de hombres y mujeres y, en contextos genéricos o inespecíficos, a personas de uno u otro sexo*».[7]

Por otra parte, en el planteamiento y realización de tareas y ejercicios, se procurará el equilibrio en cuanto a presencia de actores de ambos géneros.

12. Medidas a adoptar en caso de confinamiento por COVID-19

En caso de tener que suspender las clases presenciales por decretarse confinamiento en el marco de la pandemia del COVID-19, y mientras dure dicho confinamiento, se adoptarán las siguientes medidas.

- Los trabajos, actividades y ejercicios del apartado **TRA** se seguirán entregando a través de Internet.

  El profesor podrá requerir del alumno las pruebas que estime oportunas para garantizar la autoría del trabajo, actividad o ejercicio, incluyendo (pero no limitándose a) una entrevista personal.

- Las pruebas evaluativas del apartado **EXA** se realizarán de forma *online* en tiempo real mediante videoconferencia.

  El profesor podrá requerir del alumno las pruebas que estime oportunas para garantizar la autoría de la prueba durante y después del desarrollo de la misma, incluyendo (pero no limitándose a) el uso de webcams, micrófono, compartición de su pantalla, entrevista personal, etcétera.

[7] https://twitter.com/RAEinforma/status/1111565711653113856

Referencias

[1] *Orden de 16 de junio de 2011*, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Web (págs. 130 a 133 del BOJA nº 149 del 1 de agosto)[8].

[2] *Orden de 29 de septiembre de 2010*, por la que se regula la evaluación, certificación, acreditación y titulación académica del alumnado que cursa enseñanzas de formación profesional inicial que forma parte del sistema educativo en la Comunidad Autónoma de Andalucía (BOJA nº 202 del 15 de octubre).

[3] Instituto Nacional de Seguridad e Higiene en el Trabajo. *Instrucción básica para el trabajador usuario de pantallas de visualización de datos.*[9]

[8] http://www.juntadeandalucia.es/boja/boletines/2011/149/d/updf/d23.pdf\T1\textbackslash#page=17

[9] http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/Instruccion_Pantallas/Instruccion_basica.pdf