Programación anual del módulo profesional

Programación

Ricardo Pérez López

Curso 2024/2025

IES «Doñana», Sanlúcar de Bda.
Departamento de Informática y Comunicaciones

Jefe de Departamento: Ricardo Pérez López

Índice

1. Información general

- **Normativa de aplicación:** [1], [2], [3], [4], [5], [6], [7].
- **Equivalencia en créditos ECTS:** 15.
- **Código:** 0485.
- **Duración total:** 256 horas (32 semanas)
- **Carga lectiva semanal:** 8 horas
- Módulo no asociado a unidad de competencia.

2. Objetivos generales

1. La formación del módulo contribuye a alcanzar los **objetivos generales** de este ciclo formativo que se relacionan a continuación:

   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.

   w) Evaluar situaciones de prevención de riesgos laborales y de protección ambiental, proponiendo y aplicando medidas de prevención personales y colectivas, de acuerdo a la normativa aplicable en los procesos del trabajo, para garantizar entornos seguros.

2. La formación del módulo contribuye a alcanzar las **competencias profesionales, personales y sociales** de este título que se relacionan a continuación:

   e) Desarrollar aplicaciones web con acceso a bases de datos utilizando lenguajes, objetos de acceso y herramientas de mapeo adecuados a las especificaciones.

   j) Desarrollar e integrar componentes software en el entorno del servidor web, empleando herramientas y lenguajes específicos, para cumplir las especificaciones de la aplicación.

   t) Generar entornos seguros en el desarrollo de su trabajo y el de su equipo, supervisando y aplicando los procedimientos de prevención de riesgos laborales y ambientales de acuerdo con lo establecido por la normativa y los objetivos de la empresa.

w) Ejercer sus derechos y cumplir con las obligaciones derivadas de su actividad profesional, de acuerdo con lo establecido en la legislación vigente, participando activamente en la vida económica, social y cultural.

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:

[RA_1] Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

Criterios de evaluación:

$CE_{1.a}$) Se han identificado los bloques que componen la estructura de un programa informático.

$CE_{1.b}$) Se han utilizado entornos integrados de desarrollo.

$CE_{1.c}$) Se han creado proyectos de desarrollo de aplicaciones.

$CE_{1.d}$) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.

$CE_{1.e}$) Se ha modificado el código de un programa para crear y utilizar variables.

$CE_{1.f}$) Se han creado y utilizado constantes y literales.

$CE_{1.g}$) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.

$CE_{1.h}$) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.

$CE_{1.i}$) Se han introducido comentarios en el código.

[RA_2] Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

Criterios de evaluación:

$CE_{2.a}$) Se han identificado los fundamentos de la programación orientada a objetos.

$CE_{2.b}$) Se han instanciado objetos a partir de clases predefinidas.

$CE_{2.c}$) Se han utilizado constructores.

$CE_{2.d}$) Se han utilizado métodos y propiedades de los objetos.

$CE_{2.e}$) Se han escrito llamadas a métodos estáticos.

$CE_{2.f}$) Se han utilizado parámetros en la llamada a métodos.

$CE_{2.g}$) Se han incorporado y utilizado librerías de objetos.

$CE_{2.h}$) Se han escrito programas simples.

$CE_{2.i}$) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.

[RA_3] Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.

Criterios de evaluación:

$CE_{3.a}$) Se ha escrito y probado código que haga uso de estructuras de selección.

$CE_{3.b}$) Se han utilizado estructuras de repetición.

$CE_{3.c}$) Se han reconocido las posibilidades de las sentencias de salto.

$CE_{3.d}$) Se ha escrito código utilizando control de excepciones.

$CE_{3.e}$) Se han creado excepciones.

$CE_{3.f}$) Se han creado programas ejecutables utilizando diferentes estructuras de control.

$CE_{3.g}$) Se han probado y depurado los programas.

$CE_{3.h}$) Se han utilizado aserciones para la detección y corrección de errores durante la fase de desarrollo.

$CE_{3.i}$) Se ha comentado y documentado el código.

[RA_4] Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

Criterios de evaluación:

$CE_{4.a}$) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.

$CE_{4.b}$) Se han definido clases.

$CE_{4.c}$) Se han definido propiedades y métodos.

$CE_{4.d}$) Se han creado y utilizado métodos estáticos.

$CE_{4.e}$) Se han creado constructores.

$CE_{4.f}$) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

$CE_{4.g}$) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

$CE_{4.h}$) Se han definido y utilizado clases heredadas.

CE$_{4.i}$) Se han creado y utilizado conjuntos y librerías de clases.

[RA$_5$] Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

Criterios de evaluación:

CE$_{5.a}$) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

CE$_{5.b}$) Se han aplicado formatos en la visualización de la información.

CE$_{5.c}$) Se han reconocido las posibilidades de entrada / salida del lenguaje y las librerías asociadas.

CE$_{5.d}$) Se han utilizado ficheros para almacenar y recuperar información.

CE$_{5.e}$) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

CE$_{5.f}$) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

CE$_{5.g}$) Se han programado controladores de eventos.

CE$_{5.h}$) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

[RA$_6$] Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

Criterios de evaluación:

CE$_{6.a}$) Se han escrito programas que utilicen matrices (*arrays*).

CE$_{6.b}$) Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.

CE$_{6.c}$) Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles.

CE$_{6.d}$) Se han utilizado listas para almacenar y procesar información.

CE$_{6.e}$) Se han utilizado iteradores para recorrer los elementos de las listas.

CE$_{6.f}$) Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones.

CE$_{6.g}$) Se han creado clases y métodos genéricos.

CE$_{6.h}$) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.

CE$_{6.i}$) Se han identificado las clases relacionadas con el tratamiento de documentos escritos en diferentes lenguajes de intercambio de datos.

CE$_{6.j}$) Se han realizado programas que realicen manipulaciones sobre documentos escritos en diferentes lenguajes de intercambio de datos.

[RA$_7$] Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

Criterios de evaluación:

CE$_{7.a}$) Se han identificado y evaluado los escenarios de utilización de la herencia y la composición.

CE$_{7.b}$) Se han identificado los conceptos de herencia, superclase y subclase.

CE$_{7.c}$) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

CE$_{7.d}$) Se ha reconocido la incidencia de los constructores en la herencia.

CE$_{7.e}$) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

CE$_{7.f}$) Se han diseñado y aplicado jerarquías de clases.

CE$_{7.g}$) Se han probado y depurado las jerarquías de clases.

CE$_{7.h}$) Se han identificado y evaluado los escenarios de uso de interfaces.

CE$_{7.i}$) Se han realizado programas que implementen y utilicen jerarquías de clases.

CE$_{7.j}$) Se ha comentado y documentado el código.

[RA$_8$] Utiliza bases de datos orientadas a objetos, analizando sus características y aplicando técnicas para mantener la persistencia de la información.

Criterios de evaluación:

CE$_{8.a}$) Se han identificado las características de las bases de datos orientadas a objetos.

CE$_{8.b}$) Se ha analizado su aplicación en el desarrollo de aplicaciones mediante lenguajes orientados a objetos.

CE$_{8.c}$) Se han instalado sistemas gestores de bases de datos orientados a objetos.

CE$_{8.d}$) Se han clasificado y analizado los distintos métodos soportados por los sistemas gestores para la gestión de la información almacenada.

CE$_{8.e}$) Se han creado bases de datos y las estructuras necesarias para el almacenamiento de objetos.

CE$_{8.f}$) Se han programado aplicaciones que almacenen objetos en las bases de datos creadas.

CE$_{8.g}$) Se han realizado programas para recuperar, actualizar y eliminar objetos de las bases de datos.

CE$_{8.h}$) Se han realizado programas para almacenar y gestionar tipos de datos estructurados, compuestos y relacionados.

[RA$_9$] Gestiona información almacenada en bases de datos relacionales manteniendo la integridad y consistencia de los datos.
Criterios de evaluación:

CE$_{9.a}$) Se han identificado las características y métodos de acceso a sistemas gestores de bases de datos.

CE$_{9.b}$) Se han programado conexiones con bases de datos.

CE$_{9.c}$) Se ha escrito código para almacenar información en bases de datos.

CE$_{9.d}$) Se han creado programas para recuperar y mostrar información almacenada en bases de datos.

CE$_{9.e}$) Se han efectuado borrados y modificaciones sobre la información almacenada.

CE$_{9.f}$) Se han creado aplicaciones que muestren la información almacenada en bases de datos.

CE$_{9.g}$) Se han creado aplicaciones para gestionar la información presente en bases de datos.

4. 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*).

4.1. Cuadro resumen

Unidad didáctica	Inicio estimado
1. Introducción *#ev1*	2024-09-16
2. Expresiones *#ev1*	2024-09-23
3. Programación funcional (I) *#ev1*	2024-09-30
4. Abstracciones funcionales *#ev1*	2024-10-07
5. Programación funcional (II) *#ev1*	2024-10-14
6. Programación imperativa *#ev1*	2024-10-21
7. Programación estructurada *#ev1*	2024-10-28
8. Tipos de datos estructurados *#ev1*	2024-11-04
9. Programación modular (I) *#ev1*	2024-11-11
10. Abstracción de datos *#ev1*	2024-11-18
11. Calidad *#ev1*	2024-11-25
12. Programación orientada a objetos *#ev2*	2025-01-08
13. Relaciones entre clases *#ev2*	2025-01-15
14. Programación de interfaces gráficas de usuario *#ev2*	2025-01-22
15. Bases de datos orientadas a objetos *#ev2*	2025-01-29
16. Introducción a la tecnología Java *#ev2*	2025-02-05
17. Elementos básicos del lenguaje Java *#ev2*	2025-02-12
18. Programación orientada a objetos en Java *#ev2*	2025-02-19
19. Diseño de clases en Java *#ev2*	2025-02-26
20. Relaciones entre clases en Java *#ev2*	2025-03-05
21. Programación modular (II) *#ev2*	2025-03-12
22. Programación genérica *#ev2*	2025-03-19
23. Control de excepciones en Java *#ev2*	2025-03-26
24. Java Collections Framework (I) *#ev3*	2025-04-02
25. Java Collections Framework (II) *#ev3*	2025-04-09
26. Gestión de bases de datos relacionales *#ev3*	2025-04-23
27. Estructuras de datos lineales *#ev3 #opcional*	2025-04-30
28. Ordenación y búsqueda *#ev3 #opcional*	2025-05-07
29. Estructuras de datos no lineales *#ev3 #opcional*	2025-05-14

4.2. 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: 2024–09–16)

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 e intérpretes

1.4.1. Traductores

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

- Implementación

1.5.7. Traducción y ejecución

1.5.8. Pruebas

1.5.9. Depuración

1.5.10. Documentación

1.5.11. Mantenimiento

- Composición de operaciones y funciones

2.4.4. Métodos

2.5. Otros conceptos sobre operaciones

2.5.1. Árboles sintácticos y evaluación

2.5.2. Tipos polimórficos y operaciones polimórficas

2.5.3. Sobrecarga de operaciones

2.5.4. Equivalencia entre formas de operaciones

2.5.5. 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 #ce1d #ce1f #ce1g #ce1h #ce1i #ev1 #ra1*
(est: 2024–09–30)

3.1. Introducción

3.1.1. Concepto

3.1.2. Transparencia referencial

- Efectos laterales

3.1.3. Modelo de ejecución

- Modelo de sustitución

3.2. Tipos de datos

3.2.1. Concepto

3.2.2. Tipo de un dato

3.2.3. `type`

3.2.4. Sistemas de tipos

- Errores de tipos
- Tipado fuerte vs. débil

3.2.5. Tipos de datos básicos

- Números
- Cadenas
- Funciones

4.1.3. Aplicación funcional
- Evaluación de una aplicación funcional
- Funciones con nombre

4.1.4. Variables ligadas y libres

4.2. Ámbitos

4.2.1. Ámbitos léxicos

4.2.2. Ámbito de una definición y de una ligadura
- Visibilidad
- Tiempo de vida
- Almacenamiento

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. Ámbito de una variable libre

4.3. Evaluación

4.3.1. Entorno (*environment*)
- Ámbitos, marcos y entornos

4.3.2. Evaluación de expresiones con entornos

4.3.3. Evaluación de expresiones lambda con entornos
- Variables *sombreadas*
- Renombrado de parámetros
- Visualización en *Pythontutor*

4.3.4. Resolución de atributos de objetos

4.3.5. 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

5 **Programación funcional (II)** *#ce1a #ce1b #ce1c #ce1e #ce1g #ce2g #ce3g #ce3i #ce6f #ev1 #ra1 #ra2 #ra3 #ra6* (est: 2024–10–14)

5.1. Computabilidad

5.1.1. Funciones y procesos

5.1.2. Funciones *ad-hoc*

5.1.3. Funciones recursivas

- Definición
- Casos base y casos recursivos
- El factorial
- Diseño de funciones recursivas
  - Identificación de casos base
  - Descomposición (reducción) del problema
  - Pensamiento optimista
- Recursividad lineal
  - Procesos recursivos lineales
  - Procesos iterativos linales
- Recursividad múltiple
- Recursividad final y no final

5.1.4. La pila de control

5.1.5. Un lenguaje Turing-completo

5.2. Tipos de datos recursivos

5.2.1. Concepto

5.2.2. Cadenas

5.2.3. Tuplas

5.2.4. Rangos

5.2.5. Conversión a tupla

5.3. Funciones de orden superior

5.3.1. Concepto

5.3.2. `map`

5.3.3. `filter`

5.3.4. `reduce`

5.3.5. Expresiones generadoras

6 **Programación imperativa** *#ce1a #ce1b #ce1c #ce1e #ce2h #ce2i #ce3c #ce3g #ce3i #ce5a #ce5b #ce5c #ce5d #ce5e #ce6d #ev1 #ra1 #ra2 #ra3 #ra5 #ra6* (est: 2024–10–21)

6.1. Modelo de ejecución

6.1.1. Máquina de estados

6.4.6. Entrada y salida por archivos

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

6.5. Saltos

6.5.1. Incondicionales

6.5.2. Condicionales

7 Programación estructurada *#ce1a #ce1b #ce1c #ce2h #ce2i #ce3a #ce3b #ce3c #ce3d #ce3f #ce3g #ce3i #ce4d #ev1 #ra1 #ra2 #ra3 #ra4* (est: 2024–10–28)

7.1. Aspectos teóricos de la programación estructurada

7.1.1. Programación estructurada

7.1.2. Programa restringido

7.1.3. Programa propio

7.1.4. Estructura

7.1.5. Programa estructurado

- Ventajas de los programas estructurados

7.1.6. Teorema de Böhm-Jacopini

7.2. Estructuras básicas de control en Python

7.2.1. Secuencia

7.2.2. Selección

7.2.3. Iteración

7.2.4. Otras sentencias de control

- `break`
- `continue`
- Excepciones
  - Gestión de excepciones
- Gestores de contexto

7.3. Metodología de la programación estructurada

7.3.1. Recursos abstractos

7.3.2. Diseño descendente

7.3.3. Refinamiento sucesivo

7.4. Programación procedimental

7.4.1. Procedimientos

7.4.2. El paradigma de programación procedimental

7.4.3. Procedimientos y refinamiento sucesivo

7.4.4. Funciones imperativas

- Definición de funciones imperativas

7.4.5. Llamadas a funciones imperativas

7.4.6. Paso de argumentos

7.4.7. La sentencia `return`

7.4.8. Ámbito de variables

- Variables locales
- Variables globales
  - `global`
  - Efectos laterales

7.4.9. Funciones locales a funciones

- `nonlocal`

8 **Tipos de datos estructurados** *#ce1d #ce1h #ce2d #ce2f #ce2g #ce2h #ce2i #ce3g #ce3i #ce5b #ce6b #ce6c #ce6d #ce6e #ce6h #ce6i #ce6j #ev1 #ra1 #ra2 #ra3 #ra5 #ra6* (est: 2024–11–04)

8.1. Introducción

8.1.1. Composición

8.1.2. Conceptos básicos

8.1.3. Clasificación

8.1.4. *Hashables*

8.1.5. Iterables

8.1.6. Iteradores

- El bucle `for`
- El módulo `itertools`

8.2. Secuencias

8.2.1. Concepto de secuencia

8.2.2. Operaciones comunes

8.2.3. Inmutables

- Cadenas (`str`)
  - Formateado de cadenas
  - Expresiones regulares
- Tuplas
- Rangos

8.2.4. Mutables

- Listas
  - Listas por compresión
- Operaciones mutadoras

8.3. Estructuras no secuenciales

8.3.1. Conjuntos (`set` y `frozenset`)

- Conjuntos por comprensión
- Operaciones
- Operaciones sobre conjuntos mutables
- Recorrido de conjuntos

8.3.2. Diccionarios (`dict`)

- Diccionarios por comprensión
- Operaciones
- Recorrido de diccionarios

8.3.3. Documentos XML

- Acceso
- Modificación

9 Programación modular (I) *#ce1a #ce1b #ce1c #ce2g #ce3g #ce3i #ev1 #ra1 #ra2 #ra3* (est: 2024–11–11)

9.1. Introducción

9.1.1. Modularidad

9.1.2. Descomposición de problemas

9.1.3. Beneficios de la modularidad

9.2. Diseño modular

9.2.1. Creadores y usuarios

9.2.2. Partes de un módulo

- Interfaz
  - Especificación
- Implementación

9.2.3. Diagramas de estructura

9.3. Programación modular en Python

9.3.1. *Scripts* como módulos

9.3.2. Importación de módulos

9.3.3. Módulos como *scripts*

9.3.4. La librería estándar

13.2. Herencia

13.2.1. Generalización

13.2.2. Modos

- Herencia simple
- Visibilidad de miembros y herencia
  - Visibilidad protegida
- La clase `object`
- Herencia múltiple

13.3. Polimorfismo

13.3.1. Concepto

13.3.2. Principio de sustitución de Liskov

13.3.3. *Duck typing*

13.3.4. Sobreescritura de métodos

- Polimorfismo y métodos redefinidos

13.3.5. Ligadura dinámica

13.3.6. `super`

13.3.7. Sobreescritura de constructores

13.3.8. Clases abstractas y métodos abstractos

13.4. Herencia vs. composición

14 **Programación de interfaces gráficas de usuario** *#ce1a #ce1b #ce1c #ce2d #ce2f #ce2g #ce3f #ce3g #ce5f #ce5g #ce5h #ce6c #ce6d #ev2 #ra1 #ra2 #ra3 #ra5 #ra6* (est: 2025–01–22)

14.1. JFC y Swing

14.2. Componentes de Swing

14.3. Contenedores de nivel superior

14.3.1. `JFrame`

14.3.2. `JDialog`

14.3.3. `JApplet`

14.4. `JComponent`

14.5. Componentes de texto

14.5.1. `JTextComponent`

14.6. Marcos

14.7. Etiquetas

14.8. Botones

14.9. Arquitectura de modelos de Swing

15 **Bases de datos orientadas a objetos** *#ce8a #ce8b #ce8c #ce8d #ce8e #ce8f #ce8g #ce8h #ev2 #ra8* (est: 2025–01–29)

15.1. Introducción a las bases de datos orientadas a objetos (OODB)

15.1.1. Conceptos clave

- Objetos, clases y herencia
- Persistencia de objetos
- Integración de datos y comportamiento

15.1.2. Comparación con RDBMS

- Modelos de datos relacionales vs. orientados a objetos
- Ventajas y desventajas de OODB
- Escenarios donde es preferible usar una OODB

15.1.3. Casos de uso comunes

- Aplicaciones de simulación
- Modelos de datos complejos
- Aplicaciones con estructuras de datos jerárquicas

15.2. Persistencia de objetos en Python

15.2.1. Serialización y deserialización de objetos (módulo `pickle`)

15.3. Conceptos de bases de datos orientadas a objetos

15.3.1. Modelado de una OODB

- Clases como tablas
- Relaciones entre objetos (1 a 1, 1 a muchos, muchos a muchos)
- Polimorfismo en la base de datos

15.3.2. Implementación básica en Python

- Guardar objetos en una estructura persistente
- Simular una base de datos orientada a objetos sencilla

15.3.3. Limitaciones de una implementación casera

- Consistencia de datos
- Concurrencia
- Integridad referencial

15.4. Introducción a Zope Object Database (ZODB)

15.4.1. ¿Qué es ZODB?

- Ventajas de usar ZODB
- Comparación con otras soluciones de bases de datos

15.4.2. Instalación y configuración

- Estructura básica de un proyecto con ZODB

15.4.3. Persistencia de objetos en ZODB

- Crear una base de datos en ZODB
- Guardar, actualizar y eliminar objetos
- Administración de la base de datos y control de versiones

16 **Introducción a la tecnología Java** *#ce1a #ce1b #ce1c #ce2f #ce2g #ce2h #ce2i #ce3g #ce3i #ev2 #ra1 #ra2* (est: 2025–02–05)

16.1. Introducción

16.1.1. Historia

16.1.2. Versiones

16.1.3. Características principales

16.2. La tecnología Java

16.2.1. Máquinas reales vs. virtuales

16.2.2. Código objeto (*bytecode*)

16.2.3. La plataforma Java

- La máquina virtual de Java (JVM)
- La API de Java

16.2.4. Las herramientas de desarrollo de Java (JDK)

- El compilador `javac`
- El intérprete interactivo `jshell`

16.2.5. El entorno de ejecución de Java (JRE)

- El intérprete `java`

16.3. El primer programa Java

16.3.1. Tipado estático vs. dinámico

16.3.2. El método `main`

16.3.3. La clase principal

16.3.4. La clase `System`

16.3.5. El paquete `java.lang`

16.3.6. El objeto `out`

16.3.7. El método `println`

17 **Elementos básicos del lenguaje Java** *#ce1a #ce1b #ce1c #ce1d #ce1e #ce1f #ce1h #ce1i #ce2b #ce2c #ce2d #ce2f #ce2g #ce2i #ce3a #ce3b #ce3c #ce3e #ce3f #ce3g #ce5a #ce5c #ev2 #ra1 #ra2 #ra3 #ra5* (est: 2025–02–12)

17.1. Tipos y valores en Java

17.1.1. Introducción

17.1.2. 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

17.1.3. Tipos referencia
- Nulo
- Acceso a miembros
  - Llamadas a métodos sobrecargados

17.2. Variables en Java

17.2.1. Introducción

17.2.2. Variables de tipos primitivos

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

17.2.4. Declaración de variables
- Inicialización y asignación de variables
  - Declaración vs. definición
  - 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

17.3. Estructuras de control

17.3.1. Bloques

17.3.2. `if`

17.3.3. `switch`

18.3. *Arrays*

18.3.1. Definición

18.3.2. Declaración

18.3.3. Creación

18.3.4. Inicialización

18.3.5. Acceso a elementos

18.3.6. Longitud de un *array*

18.3.7. Modificación de elementos

18.3.8. *Arrays* de tipos referencia

18.3.9. Subtipado entre *arrays*

18.3.10. `java.util.Arrays`

18.3.11. Copia y redimensionado de *arrays*

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

18.3.12. Comparación de *arrays*

- `Arrays.equals`

18.3.13. *Arrays* multidimensionales

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

19 **Diseño de clases en Java** *#ce1a #ce1b #ce1c #ce2a #ce2b #ce2c #ce2d #ce2e #ce2f #ce2h #ce2i #ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4g #ce4h #ce4i #ev2 #ra1 #ra2 #ra4* (est: 2025–02–26)

19.1. Definición de clases

19.1.1. Sintaxis básica

19.1.2. Clases y paquetes

19.1.3. Visibilidad de una clase

- Visibilidad predeterminada
- Visibilidad pública

19.1.4. Visibilidad de un miembro de una clase

19.2. Miembros de instancia

19.2.1. Variables de instancia

- Acceso y modificación
- Variables de instancia finales

- Sobreescritura de `equals`
- Sobreescritura de `hashCode`

20.4. Restricciones

20.4.1. Clases y métodos abstractos

20.4.2. Clases y métodos finales

21 **Programación modular (II)** *#ce1a #ce1b #ce1c #ce2g #ce3g #ce3i #ce4i #ce7h #ev2 #ra1 #ra2 #ra3 #ra4 #ra7* (est: 2025–03–12)

21.1. Interfaces

21.1.1. Definición de interfaces

21.1.2. Implementación de interfaces

21.1.3. Las interfaces como tipos

21.1.4. Herencia entre interfaces

21.1.5. Métodos predeterminados

21.1.6. Ejemplo: Interfaz `CharSequence`

21.1.7. Ejemplo: Clonación de objetos

- `Cloneable`
- `Object.clone`
- Constructor de copia

21.1.8. Clases abstractas vs. interfaces

21.2. Paquetes y módulos

22 **Programación genérica** *#ce1a #ce1b #ce1c #ce3g #ce3i #ce6g #ev2 #ra1 #ra3 #ra6* (est: 2025–03–19)

22.1. Tipos genéricos

22.1.1. Parámetros de tipo

22.1.2. Argumentos de tipo

22.1.3. Tipos crudos

22.2. Métodos genéricos

22.3. Subtipos

22.3.1. Parámetros de tipo acotados

22.3.2. Clases genéricas, herencia y subtipos

- Covarianza
- Contravarianza
- Invarianza

22.4. Inferencia de tipos

22.5. Comodines

- `java.util.LinkedList`

24.5.2. Interfaz `java.util.Deque`

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

25 Java Collections Framework (II) *#ce1a #ce1b #ce1c #ce2d #ce2f #ce2g #ce3g #ce3i #ce4i #ce6a #ce6b #ce6c #ce6d #ce6e #ce6f #ev3 #ra1 #ra2 #ra3 #ra4 #ra6* (est: 2025–04–09)

25.1. Conjuntos

25.1.1. Interfaz `java.util.Set`

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

25.1.2. Interfaz `java.util.SortedSet`

- `java.util.TreeSet`

25.1.3. Interfaz `java.util.NavigableSet`

- `java.util.TreeSet`

25.2. Diccionarios

25.2.1. Interfaz `java.util.Map`

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

25.2.2. Interfaz `java.util.SortedMap`

- `java.util.TreeMap`

25.2.3. Interfaz `java.util.NavigableMap`

- `java.util.TreeMap`

26 Gestión de bases de datos relacionales *#ce1a #ce1b #ce1c #ce2d #ce2f #ce2g #ce3g #ce3i #ce6c #ce6d #ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g #ev3 #ra1 #ra2 #ra3 #ra6 #ra9* (est: 2025–04–23)

26.1. Controlador JDBC

26.1.1. Instalación

26.1.2. `CLASSPATH`

- En consola
- En Visual Studio Code

26.2. Programa de ejemplo

26.3. Ejecutar programa

26.3.1. En consola

26.3.2. En Visual Studio Code

26.4. Establecimiento de conexiones

26.5. Recuperación de información

26.5.1. Ejecución de consultas

26.5.2. Selección de registros

26.5.3. Uso de parámetros

26.6. Manipulación de la información

26.6.1. Altas, bajas y modificaciones

27 **Estructuras de datos lineales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ce6c #ce6d #ce6e #ev3 #opcional #ra1 #ra3 #ra6* (est: 2025–04–30)

27.1. Acceso secuencial

27.1.1. Listas

- Enlazadas
- Doblemente enlazadas

27.1.2. Pilas

27.1.3. Colas

27.2. Acceso directo

27.2.1. Tablas *hash*

28 **Ordenación y búsqueda** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6c #ce6d #ce6e #ev3 #opcional #ra1 #ra3 #ra6* (est: 2025–05–07)

28.1. Algoritmos de búsqueda

28.1.1. Búsqueda secuencial

28.1.2. Búsqueda dicotómica

28.2. Algoritmos de ordenación

28.2.1. Inserción directa

28.2.2. Selección directa

28.2.3. Burbuja

28.2.4. *Quicksort*

28.2.5. *Mergesort*

28.3. Tablas *Hash*

29 **Estructuras de datos no lineales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6c #ce6d #ce6e #ev3 #opcional #ra1 #ra3 #ra6* (est: 2025–05–14)

29.1. Árboles

29.1.1. Binarios

- Recorridos
  - Preorden
  - Inorden
  - Postorden

29.1.2. De búsqueda

29.1.3. Montículos

- Algoritmo de ordenación

29.1.4. Generales

- Recorrido en profundidad
- Recorrido en anchura

29.2. Grafos

29.2.1. Algoritmo de Dijkstra

29.2.2. Algoritmo de Floyd

4.3. 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	*#ce1d* *#ce1f* *#ce1g*	*#ce2e* *#ce2f* *#ce2g*							
3. Programación funcional (I)	*#ce1a* *#ce1b* *#ce1c* *#ce1d* *#ce1f* *#ce1g* *#ce1h* *#ce1i*								

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
4. Abstracciones funcionales	#ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i		#ce3g #ce3i						
5. Programación funcional (II)	#ce1a #ce1b #ce1c #ce1e #ce1g	#ce2g	#ce3g #ce3i			#ce6f			
6. Programación imperativa	#ce1a #ce1b #ce1c #ce1e	#ce2h #ce2i	#ce3c #ce3g #ce3i		#ce5a #ce5b #ce5c #ce5d #ce5e	#ce6d			
7. Programación estructurada	#ce1a #ce1b #ce1c	#ce2h #ce2i	#ce3a #ce3b #ce3c #ce3d #ce3f #ce3g #ce3i	#ce4d					
8. Tipos de datos estructurados	#ce1d #ce1h	#ce2d #ce2f #ce2g #ce2h #ce2i	#ce3g #ce3i		#ce5b	#ce6b #ce6c #ce6d #ce6e #ce6h #ce6i #ce6j			
9. Programación modular (I)	#ce1a #ce1b #ce1c	#ce2g	#ce3g #ce3i						
10. Abstracción de datos	#ce1a #ce1b #ce1c		#ce3g #ce3i			#ce6d #ce6e			
11. Calidad	#ce1a #ce1b #ce1c #ce1i	#ce2g	#ce3f #ce3g #ce3i						

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
12. Programación orientada a objetos	#ce1a #ce1b #ce1c #ce1e #ce1g	#ce2a #ce2b #ce2c #ce2d #ce2e #ce2f #ce2h #ce2i	#ce3f #ce3g #ce3h #ce3i	#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4g #ce4h					
13. Relaciones entre clases	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4g #ce4h #ce4i			#ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h #ce7i #ce7j		
14. Programación de interfaces gráficas de usuario	#ce1a #ce1b #ce1c	#ce2d #ce2f #ce2g	#ce3f #ce3g		#ce5f #ce5g #ce5h	#ce6c #ce6d			
15. Bases de datos orientadas a objetos								#ce8a #ce8b #ce8c #ce8d #ce8e #ce8f #ce8g #ce8h	
16. Introducción a la tecnología Java	#ce1a #ce1b #ce1c	#ce2f #ce2g #ce2h #ce2i							
17. Elementos básicos del lenguaje Java	#ce1a #ce1b #ce1c #ce1d #ce1e #ce1f #ce1h #ce1i	#ce2b #ce2c #ce2d #ce2f #ce2g #ce2i	#ce3a #ce3b #ce3c #ce3e #ce3f #ce3g		#ce5a #ce5c				

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
18. Programación orientada a objetos en Java	#ce1a #ce1b #ce1c	#ce2a #ce2c #ce2d #ce2e #ce2f #ce2g #ce2h #ce2i	#ce3g #ce3i			#ce6a			
19. Diseño de clases en Java	#ce1a #ce1b #ce1c	#ce2a #ce2b #ce2c #ce2d #ce2e #ce2f #ce2h #ce2i		#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4g #ce4h #ce4i					
20. Relaciones entre clases en Java	#ce1a #ce1b #ce1c		#ce3g #ce3i	#ce4f #ce4h			#ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7i #ce7j		
21. Programación modular (II)	#ce1a #ce1b #ce1c	#ce2g	#ce3g #ce3i	#ce4i			#ce7h		
22. Programación genérica	#ce1a #ce1b #ce1c		#ce3g #ce3i			#ce6g			
23. Control de excepciones en Java	#ce1a #ce1b #ce1c		#ce3d #ce3e #ce3g #ce3i						
24. Java Collections Framework (I)	#ce1a #ce1b #ce1c	#ce2d #ce2f #ce2g	#ce3g #ce3i	#ce4i		#ce6a #ce6b #ce6c #ce6d #ce6e #ce6f			

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
25. Java Collections Framework (II)	#ce1a #ce1b #ce1c	#ce2d #ce2f #ce2g	#ce3g #ce3i	#ce4i		#ce6a #ce6b #ce6c #ce6d #ce6e #ce6f			
26. Gestión de bases de datos relacionales	#ce1a #ce1b #ce1c	#ce2d #ce2f #ce2g	#ce3g #ce3i			#ce6c #ce6d			#ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g
27. Estructuras de datos lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a #ce6c #ce6d #ce6e			
28. Ordenación y búsqueda	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6c #ce6d #ce6e			
29. Estructuras de datos no lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6c #ce6d #ce6e			

5. Evaluación y calificación

La evaluación tendrá como finalidad determinar el nivel de competencia de los alumnos y la consecución de los objetivos del módulo. 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.

Durante el desarrollo del curso existirán, al menos, una evaluación inicial, tres evaluaciones parciales (al final de cada trimestre) y una evaluación final.

La evaluación se realizará atendiendo a los criterios de evaluación asociados a cada resultado de aprendizaje del módulo. Para ello, se usarán instrumentos de evaluación que medirán si el alumnado cumple con los criterios de evaluación y en qué grado.

- En cada **evaluación parcial** se medirá el grado de *cumplimiento* de los *criterios de evaluación* trabajados y evaluados en el trimestre correspondiente.

  Se entenderá que el alumno **cumple** un determinado **criterio de evaluación** si supera al menos un instrumento de evaluación que mida el grado de cumplimiento de dicho criterio de evaluación.

- En la **evaluación final** se determinará si el alumnado *alcanza*, y en qué medida, los *resultados de aprendizaje* del módulo a partir de los criterios de evaluación trabajados y evaluados que estén asociados a cada resultado de aprendizaje y en función de la ponderación que tenga cada criterio de evaluación en su correspondiente resultado de aprendizaje, según se establece en el apartado 5.3.

  Para la superación del módulo se requerirá que el alumno **alcance** todos los **resultados de aprendizaje** del módulo.

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 una prueba 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**.

5.1. Valoración general de los instrumentos de evaluación y calificación

Los instrumentos de evaluación y calificación usados durante el curso para medir el grado de cumplimiento de los criterios de evaluación se clasifican en dos grandes tipos:

- Trabajos, actividades y ejercicios (casa, clase, grupo) (tipo **TRA**).
- Pruebas evaluativas (tipo **EXA**).

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.

Los trabajos, actividades y/o ejercicios (**TRA**) versarán 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 (**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.

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 (**TEO**).

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

La calificación del examen (**EXA**), según se recoge en el algoritmo 1, se valorará de 0,0 a 10,0 y se calculará como una media ponderada de ambas partes **TEO** y **PRA**, de forma que para superar el examen será necesario obtener una calificación mínima de 3,0 en la parte teórica y de 4,0 en la parte práctica; en caso contrario, la calificación del examen no podrá ser superior a 4,0.

Algoritmo 1 (Cálculo de la calificación del examen)

$$M = 0{,}3 \cdot \textbf{TEO} + 0{,}7 \cdot \textbf{PRA}$$

$$\textbf{EXA} = \begin{cases} M & \text{si } \textbf{TEO} \geq 3{,}0 \text{ y } \textbf{PRA} \geq 4{,}0 \\ \min(M;\, 4{,}0) & \text{e. o. c.} \end{cases}$$

Donde:

M = Media ponderada entre teoría y práctica.

TEO = Calificación de la parte teórica del examen, valorada del 0,0 al 10,0.

PRA = Calificación de la parte práctica del examen, valorada del 0,0 al 10,0.

EXA = Calificación del examen, valorada del 0,0 al 10,0, calculada como la media ponderada de **TEO** y **PRA**.

5.2. Calificación

5.2.1. De un criterio de evaluación

La calificación de cada criterio de evaluación, una vez trabajado y evaluado, se valorará de 0,0 a 10,0 y se calculará como la calificación obtenida en el instrumento de evaluación y calificación usado para medir el grado de cumplimiento de ese criterio de evaluación.

Se entenderá que el alumno **cumple** un **criterio de evaluación** si la calificación que ha obtenido en el mismo es igual o superior a 4,5.

Si, a lo largo del curso, se usaran varios instrumentos para evaluar y calificar el mismo criterio de evaluación, la calificación de dicho criterio de evaluación se calcularía de la siguiente forma:

1. Se agruparán las calificaciones obtenidas en los diferentes instrumentos que hayan medido ese criterio de evaluación en función del tipo de instrumento usado para evaluarlo (**TRA** o **EXA**). Se obtendrán así dos grupos de calificaciones, uno por cada tipo de instrumento.
2. En cada grupo, se seleccionará la calificación máxima dentro de ese grupo.
3. Se calculará la media ponderada (M) de las dos calificaciones resultantes del punto anterior, atendiendo a las siguientes ponderaciones:

TRA	50 %
EXA	50 %

   Si, por algún motivo, no se usaran instrumentos de uno de los dos tipos (**TRA** o **EXA**) durante la evaluación, el otro apartado soportaría el 100 % de la carga evaluativa, de forma que la evaluación y calificación resultaría únicamente de dicho apartado.
4. Se calcula la calificación del criterio de evaluación de la siguiente forma, que garantiza que un criterio de evaluación califique como cumplido cuando haya al menos un instrumento con una calificación mayor o igual a 4,5 que mida ese criterio de evaluación:

$$\mathbf{CE}_{i\cdot j} = \begin{cases} \text{máx}(M_{ij};\ 4{,}5) & \text{si máx}\, I_{ij} \geq 4{,}5 \\ M_{ij} & \text{e. o. c.} \end{cases}$$

Donde:

$\mathbf{CE}_{i\cdot j}$ = Calificación del criterio de evaluación j asociado al resultado de aprendizaje i.

M_{ij} = Media ponderada calculada en el punto 3 anterior para el criterio de evaluación j asociado al resultado de aprendizaje i.

I_{ij} = Conjunto de todas las calificaciones obtenidas por el alumno en todos los instrumentos de evaluación y calificación que miden el criterio de evaluación j asociado al resultado de aprendizaje i.

En consecuencia:

- La calificación de un determinado criterio de evaluación podría ir cambiando a lo largo del curso en función de los instrumentos de evaluación y calificación realizados por el alumno para evaluar, recuperar, mejorar su calificación, etcétera.
- La calificación de un criterio de evaluación no podrá ser inferior a 4,5 si el alumno ha obtenido una calificación igual o superior a 4,5 en algún instrumento de evaluación y calificación que haya medido ese criterio de evaluación.

5.2.2. De un resultado de aprendizaje

La calificación de cada resultado de aprendizaje, según se recoge en el algoritmo 2, se valorará de 0,0 a 10,0 y se calculará como una media ponderada de los criterios de evaluación trabajados y evaluados que estén asociados a ese resultado de aprendizaje, siendo requisito imprescindible para alcanzar un resultado de aprendizaje el cumplir todos los criterios de evaluación trabajados y evaluados; en caso contrario, la calificación del resultado de aprendizaje no podrá ser superior a 4,0.

Algoritmo 2 (Cálculo de la calificación de un resultado de aprendizaje)

$$M_i = \frac{\sum_{j \in C_i} \mathbf{CE}_{i\cdot j} \cdot p_{ij}}{\sum_{j \in C_i} p_{ij}}$$

$$\mathbf{RA}_i = \begin{cases} M_i & \text{si } \min_{j \in C_i}\{\mathbf{CE}_{i\cdot j}\} \geq 4{,}5 \\ \min(M_i;\, 4{,}0) & \text{e. o. c.} \end{cases}$$

Donde:

M_i = Media ponderada de los criterios de evaluación asociados al resultado de aprendizaje i.

$\mathbf{RA}_i$ = Calificación del resultado de aprendizaje i.

$\mathbf{CE}_{i\cdot j}$ = Calificación del criterio de evaluación j asociado al resultado de aprendizaje i.

C_i = Conjunto de los índices (letras **a**, **b**, …) de los criterios de evaluación trabajados y evaluados, asociados al resultado de aprendizaje i.

p_{ij} = Ponderación del criterio de evaluación j en el resultado de aprendizaje i, según se determina en el apartado 5.3.

Se entenderá que el alumno ha **alcanzado** un **resultado de aprendizaje** si la calificación que ha obtenido en el mismo es igual o superior a 4,5.

5.2.3. Calificaciones parciales

La calificación del módulo en cada evaluación parcial (primer, segundo y tercer trimestres por separado) se calculará como la media ponderada de las calificaciones de los criterios de evaluación trabajados y evaluados durante el trimestre correspondiente, en función de las ponderaciones relativas indicadas en el apartado 5.3 y según se recoge en el algoritmo 3.

Algoritmo 3 (Cálculo de una calificación parcial)

$$P_i = \frac{\sum_{\mathbf{CE}_{x\cdot y} \in C_i} \mathbf{CE}_{x\cdot y} \cdot p_{xy}}{\sum_{\mathbf{CE}_{x\cdot y} \in C_i} p_{xy}}$$

Donde:

P_i = Calificación parcial del trimestre $i \in \{1,2,3\}$.

$\mathbf{CE}_{x\cdot y}$ = Calificación del criterio de evaluación y asociado al resultado de aprendizaje x.

C_i = Conjunto de los criterios de evaluación trabajados y evaluados en el trimestre i.

p_{xy} = Ponderación del criterio de evaluación y en el resultado de aprendizaje x, según se determina en el apartado 5.3.

La calificación introducida en el sistema Séneca será la calificación parcial redondeada al entero más cercano, y no podrá ser inferior a 1,0.

5.2.4. Calificación final

La calificación del módulo en la evaluación final se calculará como la media aritmética de las calificaciones de los diferentes resultados de aprendizaje, según se recoge en el algoritmo 4.

Algoritmo 4 (Cálculo de la calificación final)

$$M = \frac{\sum_{i \in R} \mathbf{RA}_i}{|R|}$$

$$F = \begin{cases} M & \text{si } \min_{i \in R}\{\mathbf{RA}_i\} \geq 4{,}5 \\ \min(M;\, 4{,}0) & \text{e. o. c.} \end{cases}$$

Donde:

M = Media aritmética de las calificaciones de los resultados de aprendizaje.

F = Calificación final del módulo.

$\mathbf{RA}_i$ = Calificación del resultado de aprendizaje i.

R = Conjunto de los índices $(1, 2, \ldots)$ de los resultados de aprendizaje del módulo.

La calificación introducida en el sistema Séneca será la calificación final redondeada al entero más cercano, y no podrá ser inferior a 1,0. Asimismo, si el alumno no ha alcanzado todos los resultados de aprendizaje, la calificación final del módulo no podrá ser superior a 4,0.

5.3. Ponderación de los criterios de evaluación

Los criterios de evaluación de cada resultado de aprendizaje se ponderan con los siguientes pesos relativos:

Resultado de aprendizaje ($\mathbf{RA}_i$)	Criterio de evaluación ($\mathbf{CE}_{i.j}$)	Porcentaje (p_{ij})
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.a}}$ *#ce1a*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.b}}$ *#ce1b*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.c}}$ *#ce1c*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.d}}$ *#ce1d*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.e}}$ *#ce1e*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.f}}$ *#ce1f*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.g}}$ *#ce1g*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.h}}$ *#ce1h*	11,1 %
$\mathbf{RA_1}$ *#ra1*	$\mathbf{CE_{1.i}}$ *#ce1i*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.a}}$ *#ce2a*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.b}}$ *#ce2b*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.c}}$ *#ce2c*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.d}}$ *#ce2d*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.e}}$ *#ce2e*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.f}}$ *#ce2f*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.g}}$ *#ce2g*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.h}}$ *#ce2h*	11,1 %
$\mathbf{RA_2}$ *#ra2*	$\mathbf{CE_{2.i}}$ *#ce2i*	11,1 %
$\mathbf{RA_3}$ *#ra3*	$\mathbf{CE_{3.a}}$ *#ce3a*	11,1 %
$\mathbf{RA_3}$ *#ra3*	$\mathbf{CE_{3.b}}$ *#ce3b*	11,1 %
$\mathbf{RA_3}$ *#ra3*	$\mathbf{CE_{3.c}}$ *#ce3c*	11,1 %
$\mathbf{RA_3}$ *#ra3*	$\mathbf{CE_{3.d}}$ *#ce3d*	11,1 %

Resultado de aprendizaje (RA_i)	Criterio de evaluación ($CE_{i.j}$)	Porcentaje (p_{ij})
RA_3 *#ra3*	$CE_{3.e}$ *#ce3e*	11,1 %
RA_3 *#ra3*	$CE_{3.f}$ *#ce3f*	11,1 %
RA_3 *#ra3*	$CE_{3.g}$ *#ce3g*	11,1 %
RA_3 *#ra3*	$CE_{3.h}$ *#ce3h*	11,1 %
RA_3 *#ra3*	$CE_{3.i}$ *#ce3i*	11,1 %
RA_4 *#ra4*	$CE_{4.a}$ *#ce4a*	11,1 %
RA_4 *#ra4*	$CE_{4.b}$ *#ce4b*	11,1 %
RA_4 *#ra4*	$CE_{4.c}$ *#ce4c*	11,1 %
RA_4 *#ra4*	$CE_{4.d}$ *#ce4d*	11,1 %
RA_4 *#ra4*	$CE_{4.e}$ *#ce4e*	11,1 %
RA_4 *#ra4*	$CE_{4.f}$ *#ce4f*	11,1 %
RA_4 *#ra4*	$CE_{4.g}$ *#ce4g*	11,1 %
RA_4 *#ra4*	$CE_{4.h}$ *#ce4h*	11,1 %
RA_4 *#ra4*	$CE_{4.i}$ *#ce4i*	11,1 %
RA_5 *#ra5*	$CE_{5.a}$ *#ce5a*	12,5 %
RA_5 *#ra5*	$CE_{5.b}$ *#ce5b*	12,5 %
RA_5 *#ra5*	$CE_{5.c}$ *#ce5c*	12,5 %
RA_5 *#ra5*	$CE_{5.d}$ *#ce5d*	12,5 %
RA_5 *#ra5*	$CE_{5.e}$ *#ce5e*	12,5 %
RA_5 *#ra5*	$CE_{5.f}$ *#ce5f*	12,5 %
RA_5 *#ra5*	$CE_{5.g}$ *#ce5g*	12,5 %
RA_5 *#ra5*	$CE_{5.h}$ *#ce5h*	12,5 %
RA_6 *#ra6*	$CE_{6.a}$ *#ce6a*	10,0 %
RA_6 *#ra6*	$CE_{6.b}$ *#ce6b*	10,0 %
RA_6 *#ra6*	$CE_{6.c}$ *#ce6c*	10,0 %
RA_6 *#ra6*	$CE_{6.d}$ *#ce6d*	10,0 %
RA_6 *#ra6*	$CE_{6.e}$ *#ce6e*	10,0 %
RA_6 *#ra6*	$CE_{6.f}$ *#ce6f*	10,0 %
RA_6 *#ra6*	$CE_{6.g}$ *#ce6g*	10,0 %
RA_6 *#ra6*	$CE_{6.h}$ *#ce6h*	10,0 %
RA_6 *#ra6*	$CE_{6.i}$ *#ce6i*	10,0 %
RA_6 *#ra6*	$CE_{6.j}$ *#ce6j*	10,0 %
RA_7 *#ra7*	$CE_{7.a}$ *#ce7a*	10,0 %
RA_7 *#ra7*	$CE_{7.b}$ *#ce7b*	10,0 %
RA_7 *#ra7*	$CE_{7.c}$ *#ce7c*	10,0 %
RA_7 *#ra7*	$CE_{7.d}$ *#ce7d*	10,0 %
RA_7 *#ra7*	$CE_{7.e}$ *#ce7e*	10,0 %
RA_7 *#ra7*	$CE_{7.f}$ *#ce7f*	10,0 %
RA_7 *#ra7*	$CE_{7.g}$ *#ce7g*	10,0 %
RA_7 *#ra7*	$CE_{7.h}$ *#ce7h*	10,0 %
RA_7 *#ra7*	$CE_{7.i}$ *#ce7i*	10,0 %

Resultado de aprendizaje (RA_i)	Criterio de evaluación ($CE_{i.j}$)	Porcentaje (p_{ij})
RA7 *#ra7*	**CE7.j** *#ce7j*	10,0 %
RA8 *#ra8*	**CE8.a** *#ce8a*	12,5 %
RA8 *#ra8*	**CE8.b** *#ce8b*	12,5 %
RA8 *#ra8*	**CE8.c** *#ce8c*	12,5 %
RA8 *#ra8*	**CE8.d** *#ce8d*	12,5 %
RA8 *#ra8*	**CE8.e** *#ce8e*	12,5 %
RA8 *#ra8*	**CE8.f** *#ce8f*	12,5 %
RA8 *#ra8*	**CE8.g** *#ce8g*	12,5 %
RA8 *#ra8*	**CE8.h** *#ce8h*	12,5 %
RA9 *#ra9*	**CE9.a** *#ce9a*	14,2 %
RA9 *#ra9*	**CE9.b** *#ce9b*	14,2 %
RA9 *#ra9*	**CE9.c** *#ce9c*	14,2 %
RA9 *#ra9*	**CE9.d** *#ce9d*	14,2 %
RA9 *#ra9*	**CE9.e** *#ce9e*	14,2 %
RA9 *#ra9*	**CE9.f** *#ce9f*	14,2 %
RA9 *#ra9*	**CE9.g** *#ce9g*	14,2 %

5.4. Medidas de recuperación y mejora de la calificación final

A lo largo del curso (idealmente, a comienzos del trimestre) se establecerán instrumentos de recuperación destinados a que el alumnado cumpla los criterios de evaluación trabajados y evaluados en trimestres anteriores que no hayan podido cumplir hasta ahora al no haber podido superar los instrumentos de evaluación correspondientes.

Los criterios de evaluación que se trabajen en unidades didácticas temporalizadas en varios trimestres podrían no necesitar de instrumentos de recuperación, ya que el alumno tendría varias oportunidades para cumplir el mismo criterio de evaluación a lo largo de varios trimestres con los instrumentos de evaluación propios de las unidades didácticas programadas para esos trimestres.

Los instrumentos de recuperación utilizados para que el alumnado pueda cumplir los criterios de evaluación que aún no ha cumplido, podrán coincidir o no con los utilizados anteriormente para evaluar el cumplimiento de tales criterios, a discreción del profesor. En particular, se podrán usar pruebas evaluativas (instrumentos del tipo **EXA**) como instrumentos de recuperación para que el alumnado pueda cumplir criterios de evaluación que aún no haya cumplido y que previamente se habían medido con instrumentos del tipo **TRA**.

Los alumnos que tengan criterios de evaluación pendientes de cumplir en la tercera evaluación, deberán presentarse directamente a la prueba final tal como se establece en el apartado

5.4.1. No hay, por tanto, pruebas de recuperación específicas para los contenidos de la tercera evaluación.

5.4.1. Prueba final

La normativa vigente establece la existencia de un examen o prueba a final de curso (usualmente en junio) para aquellos alumnos que no hayan podido superar el módulo mediante el procedimiento habitual de evaluación continua y que, por tanto, tengan pendiente alcanzar algún resultado de aprendizaje que no se haya podido alcanzar ni siquiera mediante los instrumentos de recuperación indicados anteriormente.

Asimismo, y según establece la misma normativa, esta prueba final permite al alumnado la superación del módulo incluso en el caso de no haber asistido con regularidad a clase a lo largo del curso. Es, por tanto, el único instrumento que posibilita el que un alumno que cuente con un número de faltas de asistencia superior al permitido (y que, por consiguiente, haya perdido el derecho a la evaluación continua) pueda demostrar la consecución de los objetivos del módulo y, en consecuencia, su superación.

Dicha prueba o examen final constará de varias partes separadas. De ellas, las partes que el alumno deberá realizar y superar dependerán de las calificaciones que haya obtenido en los diferentes resultados de aprendizaje a lo largo del curso, de manera que el alumno deberá realizar las partes correspondientes a los resultados de aprendizaje que aún no haya alcanzado.

Finalmente, la calificación obtenida en la prueba final se tendrá en cuenta para calcular la **calificación final del módulo** según se establece en 5.2.4.

El alumno que, debiendo presentarse a la prueba final, no lo haga, será calificado como «No evaluado» (NE).

5.4.2. Mejora de la calificación final

El alumno que, habiendo superado el módulo mediante el procedimiento habitual de evaluación continua, desee mejorar su calificación final («subir nota»), podrá hacerlo en la prueba final, manteniéndose la calificación final más alta de entre las obtenidas antes y después de presentarse a subir nota en dicha prueba final. A tal efecto, se indicará en la prueba final qué partes deberá realizar el alumnado que se haya presentado a subir nota.

6. Evaluación de la práctica docente

Se realizarán entre el alumnado tres encuestas de valoración trimestrales y anónimas, al final de cada trimestre, formadas por preguntas cuantitativas y cualitativas relacionadas con as-

pectos que afecten al módulo, su organización, su evaluación, los materiales empleados, el día a día de su práctica docente y el profesorado encargado de su docencia. A lo largo del curso se analizará la evolución de los resultados obtenidos globalmente y por cada pregunta para observar su progresión positiva o negativa y así poder tomar las acciones que se estime oportunas llegado el caso.

7. Orientaciones pedagógicas

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 prueba, depuración y documentación de los programas desarrollados.

8. 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.

Al disponer de ocho horas a la semana, repartidas entre cuatro días a dos horas por día, el objetivo en todo momento será usar los dos primeros días para exponer contenidos y resolver dudas, y los otros dos días para realizar ejemplos y ejercicios en clase. Para ello, será necesario que el alumno previamente haya trabajado en casa los contenidos que se vayan a impartir en esa semana, usando los materiales y apuntes que el profesor pone previamente a su disposición.

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.

9. Recursos

9.1. 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.

9.2. Software

- Sistema operativo GNU/Linux.

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

- El resto de herramientas y aplicaciones necesarias se instalarán a través de Internet a lo largo del curso.

9.3. 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**[3] como herramienta centralizada para compartir código y para la gestión integral de todo elemento satélite del mismo (control de versiones, desarrollo colaborativo, incidencias, etcétera).

9.4. Bibliografía

9.4.1. Principal

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

9.4.2. 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.
- 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.

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

10. 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*).

11. 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 [8].

- **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.

12. 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.

12.1. 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: Día Internacional de la Eliminación de la Violencia contra la Mujer.
   b) 30 de enero: Resolución de conflictos de forma pacífica. Día de la Paz.
   c) 8 de marzo: Día Internacional de la Mujer.

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.

13. Funciones del profesor de desdoble

Las funciones asignadas al profesor de desdoble serán las siguientes:

- Solventar dudas del alumnado sobre las actividades que se desarrollarán durante el curso.
- Impartir clases de refuerzo a los alumnos que se incorporan más tarde o necesitan un mayor esfuerzo para asimilar los conocimientos impartidos.
- Mantenimiento de los equipos informáticos o cualquier problema técnico que pueda surgir durante las clases.
- Si el profesor titular faltase, tomará el control de las clases de la tal forma que continúen con la mayor normalidad posible realizando las tareas que correspondan, bien sea impartición de clases, corrección de actividades o realización de exámenes.

  El cuaderno del profesor estará compartido para que ambos profesores puedan acceder a él sin problemas.

Referencias

[1] Real Decreto 1147/2011, de 29 de julio (BOE n.º 182 de 30 de julio) por el que se establece la ordenación general de la formación profesional del sistema educativo.

[2] Real Decreto 686/2010, de 20 de mayo (BOE n.º 143 de 12 de junio), por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Web.

[3] 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].

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

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

[4] 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).

[5] Ley Orgánica 3/2022, de 31 de marzo (BOE n.º 78 de 1 de abril), de ordenación e integración de la Formación Profesional.

[6] Real Decreto 405/2023, de 29 de mayo (BOE n.º 132 de 3 de junio), por el que se actualizan los títulos de la formación profesional del sistema educativo de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma y Técnico Superior en Desarrollo de Aplicaciones Web, de la familia profesional Informática y Comunicaciones, y se fijan sus enseñanzas mínimas.

[7] Real Decreto 659/2023, de 18 de julio (BOE n.º 174 de 22 de julio), por el que se desarrolla la ordenación del Sistema de Formación Profesional.

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

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