

 	MODELO DE PROGRAMACIÓN ANUAL	MD8101	
		VERSIÓN 0	Pág. 1 de 57

Programación anual del módulo profesional

Programación

Ricardo Pérez López

Curso 2019/2020

Departamento de Informática y Comunicaciones

Jefe de Departamento: Ricardo Pérez López

Índice

1. Información general	4
2. Objetivos generales	4
3. Resultados de aprendizaje y criterios de evaluación	5
4. Instrumentos y procedimientos de evaluación y calificación	9
4.1. Valoración general de los contenidos	9
4.2. Calificación	10
4.2.1. Calificaciones parciales	11
4.2.2. Calificación final	12
4.3. Medidas de recuperación y mejora de las calificaciones	12
4.3.1. Recuperación del apartado TRA	12
4.3.2. Recuperación del apartado EXA	13
4.3.3. Mejora de las calificaciones	13
5. Contenidos y temporalización	13
5.1. Cuadro resumen	13
5.2. Esquema detalle	14
5.3. Correspondencia con resultados de aprendizaje y criterios de evaluación	37
6. Orientaciones pedagógicas	40
7. Orientaciones metodológicas	41
8. Recursos	42
8.1. Hardware	42
8.2. Software	42
8.3. Online	43
8.4. Bibliografía	43
8.4.1. Principal	43
8.4.2. Complementaria	43
9. Atención a la diversidad	44
10. Temas transversales	44
11. Actuaciones para desarrollar la perspectiva de género	45
11.1. Actuaciones generales permanentes	46

12. Modificaciones en el marco del COVID-19	47
12.1. Resultados de aprendizaje y criterios de evaluación	47
12.1.1. Resultados de aprendizaje y criterios de evaluación ya trabajados durante los dos primeros trimestres	48
12.1.2. Resultados de aprendizaje y criterios de evaluación a trabajar durante el tercer trimestre	50
12.1.3. Resultados de aprendizaje y criterios de evaluación que no podrán trabajarse durante el presente curso	51
12.2. Contenidos y temporalización	52
12.2.1. Contenidos ya trabajados durante los dos primeros trimestres	53
12.2.2. Contenidos a trabajar durante el tercer trimestre	53
12.2.3. Contenidos que no podrán trabajarse durante el presente curso	53
12.3. Metodología	54
12.4. Actividades	54
12.5. Evaluación	55
12.5.1. Recuperación de los dos primeros trimestres	55
12.5.2. Evaluación del tercer trimestre	55
12.5.3. Evaluación final	55
12.6. Calificación	55
12.6.1. Calificaciones parciales del primer y segundo trimestres	55
12.6.2. Calificación parcial del tercer trimestre	56
12.6.3. Calificación final	56

 	MODELO DE PROGRAMACIÓN ANUAL	MD8101	
		VERSIÓN 0	Pág. 4 de 57

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.

 	MODELO DE PROGRAMACIÓN ANUAL	MD8101	
		VERSIÓN 0	Pág. 7 de 57

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.

 Junta de Andalucía		MODELO DE PROGRAMACIÓN ANUAL		MD8101	
				VERSIÓN 0	Pág. 8 de 57

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 la plataforma Ágora** 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 se calculará como una media ponderada de ambas partes (como se recoge en el siguiente epígrafe) y para superar el examen en su conjunto será necesario obtener una calificación mínima de 3 puntos en la parte teórica y de 4 en la parte práctica.

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  $\geq$  3 and PRA  $\geq$  4:
    EXA = TEO * 0.3 + PRA * 0.7
else:
    EXA = min(TEO, PRA)

if TRA  $\geq$  3 and EXA  $\geq$  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:

$$\text{NOTA} = \text{EXA}$$

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:

$$\text{NOTA} = \text{TRA}$$

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

La calificación final del módulo se calculará de la siguiente forma:

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:

EV₁: Calificación de la primera evaluación.

EV₂: Calificación de la segunda evaluación.

EV₃: Calificación de la tercera evaluación.

El módulo se considera aprobado si **NOTA** ≥ 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 <i>#ev1</i>	2019-09-17
2. Programación funcional I <i>#ev1</i>	2019-09-23
3. Programación funcional II <i>#ev1</i>	2019-09-30

Unidad didáctica	Inicio estimado
4. Programación imperativa <i>#ev1</i>	2019-10-07
5. Programación estructurada <i>#ev1</i>	2019-10-14
6. Tipos de datos estructurados <i>#ev1</i>	2019-10-21
7. Metodología de la programación <i>#ev1 #opcional</i>	2019-10-28
8. Programación modular I <i>#ev1</i>	2019-11-04
9. Abstracción de datos <i>#ev1</i>	2019-11-11
10. Entrada y salida de información <i>#ev1</i>	2019-11-18
11. Calidad I <i>#ev1</i>	2019-11-25
12. Complejidad algorítmica <i>#ev1 #opcional</i>	2019-12-02
13. Programación orientada a objetos <i>#ev2</i>	2020-01-08
14. Relaciones entre clases Python <i>#ev2</i>	2020-01-15
15. Introducción a la tecnología Java <i>#ev2</i>	2020-01-22
16. Elementos básicos del lenguaje Java <i>#ev2</i>	2020-01-29
17. Programación orientada a objetos en Java <i>#ev2</i>	2020-02-05
18. Relaciones entre clases Java <i>#ev2</i>	2020-02-12
19. Clases y objetos básicos en Java <i>#ev2</i>	2020-02-19
20. Programación modular II <i>#ev2</i>	2020-02-26
21. Programación genérica <i>#ev2</i>	2020-03-04
22. Estructuras de datos lineales <i>#ev2</i>	2020-03-11
23. Ordenación y búsqueda <i>#ev2</i>	2020-03-18
24. Estructuras de datos no lineales <i>#ev2</i>	2020-03-25
25. Control de excepciones <i>#ev2</i>	2020-04-01
26. Java Collections Framework <i>#ev3</i>	2020-04-15
27. Principios y patrones de diseño <i>#ev3 #opcional</i>	2020-04-22
28. Calidad II <i>#ev3</i>	2020-04-29
29. Gestión de bases de datos relacionales <i>#ev3</i>	2020-05-06
30. Programación de interfaces gráficas de usuario <i>#ev3 #opcional</i>	2020-05-13

5.2. Esquema detalle

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: 2019-09-17)

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 **PROGRAMACIÓN FUNCIONAL I** #ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i #ce3f #ce3g #ev1 #ra1 #ra3 #ra6 (est: 2019-09-23)

2.1. El lenguaje de programación Python

2.1.1. Historia

2.1.2. Características principales

2.2. Modelo de ejecución

2.2.1. Modelo de sustitución

2.3. Expresiones

2.3.1. Concepto

2.3.2. Evaluación de expresiones

- Valores, expresión canónica y forma normal
- Formas normales y evaluación
- Transparencia referencial

2.3.3. Literales

2.3.4. Operaciones, operadores y operandos

- Aridad de operadores
- Paréntesis
- Prioridad de operadores
- Asociatividad de operadores

2.3.5. Funciones y métodos

- Funciones
- Igualdad de funciones
- Funciones con varios argumentos
- Composición de funciones
- Métodos

2.4. Tipos de datos

2.4.1. Concepto

2.4.2. type

2.4.3. Sistemas de tipos

- Errores de tipos
- Tipado fuerte vs. débil

2.4.4. Tipos de datos básicos

- Números
- Cadenas

2.4.5. Conversión de tipos

2.4.6. Operaciones predefinidas

- Operadores predefinidos
 - Operadores aritméticos
 - Operadores de cadenas
- Funciones predefinidas
 - Funciones matemáticas
- Métodos predefinidos

2.5. Álgebra de Boole

2.5.1. El tipo de dato *booleano*

2.5.2. Operadores relacionales

2.5.3. Operadores lógicos

- Tablas de verdad

2.5.4. Axiomas

- Traducción a Python

2.5.5. Teoremas fundamentales

- Traducción a Python

2.5.6. El operador ternario

2.6. Definiciones

2.6.1. Introducción

2.6.2. Identificadores y ligaduras (*binding*)

- Reglas léxicas
- Tipo de un identificador

2.6.3. Evaluación de expresiones con ligaduras

2.6.4. Marcos (*frames*)

2.6.5. Entorno (*environment*)

2.6.6. *Scripts*

2.6.7. Ámbitos

- Ámbito de una ligadura

2.7. Documentación interna

2.7.1. Identificadores significativos

2.7.2. Comentarios

3 PROGRAMACIÓN FUNCIONAL II *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019-09-30)

3.1. Abstracciones funcionales

3.1.1. Expresiones lambda

- Parámetros y argumentos
- Aplicación funcional
 - Llamadas a funciones
 - Evaluación de una aplicación funcional
- Variables ligadas y libres
- Ámbitos
 - Ámbito de una variable ligada
 - Ámbitos, marcos y entornos
 - Variables *sombreadas*
 - Renombrado de parámetros
 - Expresiones lambda y entornos
 - Evaluación de expresiones lambda con entornos
- Pureza

3.1.2. Estrategias de evaluación

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

3.1.3. Composición de funciones

3.1.4. Las funciones como abstracciones

3.2. Computabilidad

3.2.1. Funciones y procesos

3.2.2. 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 lineales recursivos

- Procesos lineales iterativos
- Recursividad en árbol
- 3.2.3. Pila de control
- 3.2.4. Un lenguaje Turing-completo
- 3.3. Tipos de datos recursivos
 - 3.3.1. Cadenas
 - 3.3.2. Listas
 - 3.3.3. Rangos
 - 3.3.4. Conversión a lista
- 3.4. Funciones de orden superior
 - 3.4.1. Concepto
 - 3.4.2. map
 - 3.4.3. filter
 - 3.4.4. reduce
 - 3.4.5. Listas por comprensión

4 PROGRAMACIÓN IMPERATIVA *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–10–07)

- 4.1. Modelo de ejecución
 - 4.1.1. Máquina de estados
 - 4.1.2. Secuencia de instrucciones
 - 4.1.3. Sentencias
- 4.2. Asignación destructiva
 - 4.2.1. Referencias al montículo
 - 4.2.2. Variables
 - 4.2.3. Estado
 - 4.2.4. Marcos en programación imperativa
 - 4.2.5. Sentencia de asignación
 - 4.2.6. Evaluación de expresiones con variables
 - 4.2.7. Constantes
 - 4.2.8. Tipado estático vs. dinámico
 - 4.2.9. Asignación compuesta
 - 4.2.10. Asignación múltiple
- 4.3. Mutabilidad
 - 4.3.1. Estado de un dato
 - 4.3.2. Tipos mutables e inmutables
 - Inmutables
 - Mutables

4.3.3. Alias de variables

- id
- is

4.4. Cambios de estado ocultos

4.4.1. Funciones puras

4.4.2. Funciones impuras

4.4.3. Efectos laterales

4.4.4. Transparencia referencial

4.4.5. Entrada y salida por consola

- print
 - El valor None
- input

4.5. Saltos

4.5.1. Incondicionales

4.5.2. Condicionales

5 PROGRAMACIÓN ESTRUCTURADA #ce1a #ce1b #ce1c #ce3a #ce3f #ce3g #ev1 #ra1 #ra3 #ra6 (est: 2019–10–14)

5.1. Aspectos teóricos de la programación estructurada

5.1.1. Programación estructurada

5.1.2. Programa restringido

5.1.3. Programa propio

5.1.4. Estructura

5.1.5. Programa estructurado

- Ventajas de los programas estructurados

5.1.6. Teorema de Böhm-Jacopini

5.2. Estructuras básicas de control

5.2.1. Secuencia

5.2.2. Selección

5.2.3. Iteración

5.2.4. Otras sentencias de control

- break
- continue
- Excepciones
 - Gestión de excepciones

5.3. Metodología de la programación estructurada

5.3.1. Recursos abstractos

5.3.2. Diseño descendente

5.3.3. Refinamiento sucesivo

5.4. Funciones imperativas

5.4.1. Definición de funciones imperativas

5.4.2. Llamadas a funciones imperativas

5.4.3. Paso de argumentos

5.4.4. La sentencia return

5.4.5. Ámbito de variables

- Variables locales
- Variables globales
 - global
 - Efectos laterales

5.4.6. Funciones locales a funciones

- nonlocal

5.4.7. *Docstrings*

6 TIPOS DE DATOS ESTRUCTURADOS #ce1d #ce1h #ce3f #ce3g #ce5b #ce6g #ev1 #ra1 #ra3 #ra6 (est: 2019-10-21)

6.1. Conceptos básicos

6.1.1. Introducción

6.1.2. *Hashables*

6.1.3. Iterables

6.1.4. Iteradores

- Expresiones generadoras
- El bucle for
- El módulo *itertools*

6.2. Secuencias

6.2.1. Concepto de secuencia

6.2.2. Operaciones comunes

6.2.3. Inmutables

- Cadenas (*str*)
 - Formateado de cadenas
 - Expresiones regulares

- Tuplas

- Rangos

6.2.4. Mutables

- Listas

6.3. Estructuras no secuenciales

6.3.1. Conjuntos (set y frozenset)

- Operaciones
- Operaciones sobre conjuntos mutables

6.3.2. Diccionarios (dict)

- Operaciones

7 METODOLOGÍA DE LA PROGRAMACIÓN *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #opcional #ra1 #ra3 #ra6* (est: 2019–10–28)

7.1. Ciclo de vida

7.2. Especificación e implementación

7.3. Verificación y validación de programas

7.3.1. Demostraciones por inducción

7.4. Programación funcional

7.4.1. Especificaciones formales

- Como cálculo

7.4.2. Derivación de programas

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

7.5. Programación imperativa

7.5.1. Especificaciones formales

- Como modificación de estados

7.5.2. Derivación de programas

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

7.6. El lenguaje Dafny *#opcional*

8 PROGRAMACIÓN MODULAR I *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–11–04)

8.1. Introducción

8.1.1. Modularidad

8.1.2. Descomposición de problemas

8.1.3. Beneficios de la modularidad

8.2. Diseño modular

8.2.1. Partes de un módulo

- Interfaz
- Implementación

8.2.2. Diagramas de estructura

8.2.3. Programación modular en Python

- Importación de módulos
- Módulos como *scripts*
- Paquetes *#opcional*
- Documentación interna *#opcional*

8.3. Criterios de descomposición modular

8.3.1. Abstracción

8.3.2. Ocultación de información

8.3.3. Independencia funcional

- Cohesión
- Acoplamiento

8.3.4. Reusabilidad

9 ABSTRACCIÓN DE DATOS *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–11–11)

9.1. Introducción

9.1.1. Introducción

9.1.2. Tipos abstractos de datos

9.2. Especificaciones

9.2.1. Sintaxis

9.2.2. Operaciones

9.2.3. Ejemplos

9.3. Implementaciones

9.3.1. Implementaciones

9.4. Niveles y barreras de abstracción

9.4.1. Niveles de abstracción

9.4.2. Barreras de abstracción

9.4.3. Propiedades de los datos

9.5. Las funciones como datos

9.5.1. Representación funcional

9.5.2. Estado interno

9.5.3. Paso de mensajes

9.6. Abstracción de datos y modularidad

9.6.1. El tipo abstracto como módulo

10 ENTRADA Y SALIDA DE INFORMACIÓN *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5a #ce5b #ce5c #ce5d #ce5e #ce6h #ce6i #ev1 #ra1 #ra3 #ra5 #ra6* (est: 2019-11-18)

10.1. Formateado de cadenas

10.2. La consola

10.2.1. Entrada desde teclado

10.2.2. Salida a pantalla

10.3. Archivos de datos

10.3.1. Registros

10.3.2. Apertura y cierre de archivos

10.3.3. Gestores de contexto

10.3.4. Modos de acceso

10.3.5. Lectura y escritura de información en archivos

10.4. Manipulación de archivos XML

10.5. Serialización de objetos

10.6. Sistemas de archivos

10.6.1. Manipulación de los sistemas de archivos

10.6.2. Creación y eliminación de archivos y directorios

11 CALIDAD I *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3* (est: 2019-11-25)

11.1. Depuración

11.1.1. print()

11.1.2. Depuración en el IDE

11.2. Pruebas

11.2.1. Enfoques de pruebas

- Pruebas de caja blanca
- Pruebas de caja negra

11.2.2. Estrategias de pruebas

- Unitarias
- Funcionales
- De aceptación

11.2.3. doctest

11.2.4. pytest

11.2.5. Desarrollo conducido por pruebas

- Ciclo de desarrollo

- Ventajas

11.3. Documentación

11.3.1. Interna

- Comentarios
- *Docstrings*
- Reglas de estilo

11.3.2. Externa

- pydoc

12 COMPLEJIDAD ALGORÍTMICA #ev1 #opcional (est: 2019-12-02)

12.1. Introducción

12.2. Principio de invarianza

12.3. La notación asintótica $O(f(n))$

12.4. Órdenes de complejidad

12.5. Operaciones entre órdenes de complejidad

12.5.1. Regla de la suma

12.5.2. Regla del producto

12.6. Reglas prácticas para el cálculo de la eficiencia

12.7. Resolución de recurrencias

12.7.1. Reducción de problemas mediante sustracción

12.7.2. Reducción de problemas mediante división

13 PROGRAMACIÓN ORIENTADA A OBJETOS #ce1a #ce1b #ce1c #ce2a #ce2b #ce2c #ce2d #ce2f #ce2h #ce2i #ce3f #ce3g #ce6a #ev2 #ra1 #ra2 #ra3 #ra6 (est: 2020-01-08)

13.1. Introducción

13.1.1. Perspectiva histórica

13.1.2. Lenguajes orientados a objetos

13.2. Clase

13.3. Objeto

13.3.1. Referencias

13.3.2. La antisimetría dato-objeto

13.4. Estado

13.4.1. Atributos

13.5. Paso de mensajes

13.5.1. Ejecución de métodos

13.5.2. Definición de métodos

13.5.3. Métodos *mágicos* y constructores

13.6. Identidad

13.6.1. Igualdad

13.7. Encapsulación

13.8. Variables de clase

13.9. Métodos estáticos

14 RELACIONES ENTRE CLASES PYTHON *#ce1a #ce1b #ce1c #ce3f #ce3g #ce4g #ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h #ev2 #ra1 #ra3 #ra4 #ra7* (est: 2020-01-15)

14.1. Asociaciones básicas

14.1.1. Agregación

14.1.2. Composición

14.2. Herencia

14.2.1. Concepto de herencia

14.2.2. Modos

- Simple
- Múltiple

14.2.3. Superclases y subclases

14.2.4. Utilización de clases heredadas

14.3. Polimorfismo

14.3.1. Sobreescritura de métodos

14.3.2. `super()`

14.3.3. Sobreescritura de constructores

14.4. Herencia vs. composición

15 INTRODUCCIÓN A LA TECNOLOGÍA JAVA *#ce1a #ce1b #ce1c #ce1e #ce1f #ce2b #ce2i #ce3f #ce3g #ev2 #ra1 #ra2* (est: 2020-01-22)

15.1. Introducción

15.2. Compilación vs. interpretación

15.2.1. Máquinas reales vs. virtuales

15.2.2. Código objeto, *bytecode* y archivos `.class`

15.2.3. La plataforma Java

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

15.2.4. El entorno de ejecución de Java (JRE)

- El intérprete java

15.2.5. Las herramientas de desarrollo de Java (JDK)

- El compilador javac
- El intérprete interactivo jshell

15.3. Características de Java

15.4. Tipado estático vs. dinámico

15.5. El primer programa Java

15.5.1. El método main()

15.5.2. La clase System

15.5.3. El objeto out

15.5.4. El método println()

15.5.5. Compilación y ejecución en consola y en el IDE

16 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: 2020-01-29)

16.1. Tipos y valores en Java

16.1.1. Tipos primitivos

- Integrales
- De coma flotante
- Booleanos
- Conversiones entre datos primitivos
 - Casting
 - De ampliación (*widening*)
 - De restricción (*narrowing*)
- Promociones numéricas

16.1.2. Tipos por referencia

16.2. Variables en Java

16.2.1. Variables de tipos primitivos

16.2.2. Variables de tipos por referencia

16.2.3. Declaraciones de variables

16.3. Estructuras de control

16.3.1. Bloques

16.3.2. if

16.3.3. switch

16.3.4. while

16.3.5. for

16.3.6. do ... while

16.4. Entrada/salida

16.4.1. Flujos System.in, System.out y System.err

16.4.2. java.util.Scanner

17 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: 2020-02-05)

17.1. Uso básico de objetos

17.1.1. Instanciación

- new
- getClass()
- instanceof

17.1.2. Referencias

- null

17.1.3. Comparación de objetos

- equals()
- hashCode()

17.1.4. Destrucción de objetos

- Recolección de basura

17.2. Encapsulación

17.2.1. Visibilidad

- Pública
- Privada
- Por defecto

17.3. Miembros no estáticos

17.3.1. Atributos

- Acceso y modificación
- Atributos finales

17.3.2. Métodos

- Referencia this
- Constructores y destructores
- Accesos y mutadores
- Sobrecarga
- Ámbito de un identificador
- Resolución de identificadores

17.4. Miembros estáticos

17.4.1. Métodos estáticos

17.4.2. Atributos estáticos

17.4.3. Atributos estáticos finales

18 RELACIONES ENTRE CLASES JAVA #ce1a #ce1b #ce1c #ce3f #ce3g #ce4g #ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h #ev2 #ra1 #ra3 #ra4 #ra7 (est: 2020-02-12)

18.1. Asociaciones básicas

18.1.1. Agregación

18.1.2. Composición

18.2. Herencia

18.2.1. Subtipado

18.2.2. La clase Object

18.2.3. Visibilidad protegida

18.3. Polimorfismo

18.3.1. El principio de sustitución de Liskov

18.3.2. Conversiones entre tipos referencia

- *Widening*
- *Narrowing*

18.3.3. Sobreescritura de métodos

- super
- Covarianza en el tipo de retorno
- Invarianza en el tipo de los argumentos
- Sobreescritura de constructores
- Sobreescritura de equals()
- Sobreescritura de hashCode()

18.4. Restricciones

18.4.1. Clases y métodos abstractos

18.4.2. Clases y métodos finales

19 CLASES Y OBJETOS BÁSICOS EN JAVA #ce1a #ce1b #ce1c #ce2e #ce3f #ce3g #ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h #ev2 #ra1 #ra2 #ra3 #ra4 (est: 2020-02-19)

19.1. Clases *wrapper*

19.1.1. Conversiones de empaquetado/desempaquetado (*boxing/unboxing*)

19.2. Cadenas

19.2.1. Inmutables

- String

19.2.2. Mutables

- StringBuffer

- `StringBuilder`
- `StringTokenizer`

19.2.3. Conversión a `String`

19.2.4. Comparación de cadenas

19.2.5. Diferencias entre literales cadena y objetos `String`

19.3. *Arrays*

19.3.1. De tipos primitivos

- Declaración
- Creación
- Inicialización

19.3.2. `.length`

19.3.3. De objetos

- Declaración
- Creación
- Inicialización

19.3.4. `java.util.Arrays`

19.3.5. Copia y redimensionado de arrays

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

19.3.6. Comparación de *arrays*

- `Arrays.equals()`

19.3.7. Arrays multidimensionales

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

20 **PROGRAMACIÓN MODULAR II** #ce1a #ce1b #ce1c #ce3f #ce3g #ce4i #ce4j #ev2 #ra1 #ra3 #ra4 (est: 2020-02-26)

20.1. Las clases como módulos

20.1.1. Interfaz de una clase

20.1.2. Métodos *getter* y *setter*

20.2. Interfaces

20.2.1. Definición de interfaces

20.2.2. Implementación de interfaces

- 20.2.3. Las interfaces como tipos
- 20.2.4. Conversiones entre interfaces
- 20.2.5. Métodos predeterminados
- 20.2.6. Ejemplo: Interfaz CharSequence
- 20.2.7. Ejemplo: Clonación de objetos
 - Cloneable
 - Object.clone()
 - Constructor de copia

20.3. Paquetes y módulos

21 PROGRAMACIÓN GENÉRICA #ce1a #ce1b #ce1c #ce3f #ce3g #ce6f #ev2 #ra1 #ra3 #ra6 (est: 2020-03-04)

- 21.1. Tipos genéricos
 - 21.1.1. Parámetros de tipo
 - 21.1.2. Argumentos de tipo
 - 21.1.3. Tipos crudos
- 21.2. Métodos genéricos
- 21.3. Subtipos
 - 21.3.1. Parámetros de tipo acotados
 - 21.3.2. Clases genéricas, herencia y subtipos
 - Covarianza
 - Contravarianza
 - Invarianza
- 21.4. Inferencia de tipos
- 21.5. Comodines
- 21.6. Borrado de tipos
- 21.7. Limitaciones

22 ESTRUCTURAS DE DATOS LINEALES #ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ev2 #ra1 #ra3 #ra6 (est: 2020-03-11)

- 22.1. Acceso secuencial
 - 22.1.1. Listas
 - Enlazadas
 - Doblemente enlazadas
 - 22.1.2. Pilas
 - 22.1.3. Colas

22.2. Acceso directo

22.2.1. Tablas *hash*

23 ORDENACIÓN Y BÚSQUEDA *#ce1a #ce1b #ce1c #ce3f #ce3g #ev2 #ra1 #ra3 #ra6* (est: 2020-03-18)

23.1. Algoritmos de búsqueda

23.1.1. Búsqueda secuencial

23.1.2. Búsqueda dicotómica

23.2. Algoritmos de ordenación

23.2.1. Inserción directa

23.2.2. Selección directa

23.2.3. Burbuja

23.2.4. *Quicksort*

23.2.5. *Mergesort*

23.3. Tablas *Hash*

24 ESTRUCTURAS DE DATOS NO LINEALES *#ce1a #ce1b #ce1c #ce3f #ce3g #ev2 #ra1 #ra3 #ra6* (est: 2020-03-25)

24.1. Árboles

24.1.1. Binarios

- Recorridos
 - Preorden
 - Inorden
 - Postorden

24.1.2. De búsqueda

24.1.3. Montículos

- Algoritmo de ordenación

24.1.4. Generales

- Recorrido en profundidad
- Recorrido en anchura

24.2. Grafos

24.2.1. Algoritmo de Dijkstra

24.2.2. Algoritmo de Floyd

25 CONTROL DE EXCEPCIONES *#ce1a #ce1b #ce1c #ce3d #ce3f #ce3g #ev2 #ra1 #ra3* (est: 2020-04-01)

25.1. Errores y excepciones

25.2. El requisito «*captura o especifica*»

25.2.1. Tipos de excepciones

25.3. Captura y manejo de excepciones

25.3.1. Bloque try

25.3.2. Bloques catch

25.3.3. Bloque finally

25.4. Excepciones y firmas

25.5. Lanzamiento de excepciones

25.5.1. Excepciones encadenadas

25.5.2. Creación de clases de excepción

25.6. Excepciones no chequeadas

25.7. Ventajas de las excepciones

26 **JAVA COLLECTIONS FRAMEWORK** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ce6b #ce6c #ce6d #ce6e #ce6f #ev3 #ra1 #ra3 #ra6* (est: 2020-04-15)

26.1. Colecciones y arrays

26.2. Arquitectura

26.3. Tipos de colecciones

26.3.1. Listas ordenadas

26.3.2. Conjuntos

26.3.3. Diccionarios

26.4. Listas

26.4.1. java.util.List

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

26.5. Colas

26.5.1. Interfaz java.util.Queue

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

26.5.2. Interfaz java.util.Deque

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

26.6. Conjuntos

26.6.1. Interfaz java.util.Set

- java.util.HashSet

- java.util.LinkedHashSet
- java.util.TreeSet

26.6.2. Interfaz java.util.SortedSet

- java.util.TreeSet

26.6.3. Interfaz java.util.NavigableSet

- java.util.TreeSet

26.7. Diccionarios

26.7.1. Interfaz java.util.Map

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

26.7.2. Interfaz java.util.SortedMap

- java.util.TreeMap

26.7.3. Interfaz java.util.NavigableMap

- java.util.TreeMap

27 PRINCIPIOS Y PATRONES DE DISEÑO #ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #opcional #ra1 #ra3 (est: 2020-04-22)

27.1. Principios de diseño

27.1.1. Encapsulación y ocultación de información

27.1.2. Diseño orientado a interfaces

27.1.3. 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

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

27.2. Patrones de diseño

27.2.1. De creación

27.2.2. Estructurales

27.2.3. De comportamiento

28 CALIDAD II #ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3 (est: 2020-04-29)

28.1. Pruebas automáticas

28.1.1. JUnit

28.2. Documentación

28.2.1. Interna

- Reglas de estilo
- Google Java Format

28.2.2. Externa

- Javadoc

29 **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: 2020-05-06)

29.1. Controlador JDBC

29.1.1. Instalación

29.1.2. CLASSPATH

29.1.3. Carga

29.2. Establecimiento de conexiones

29.3. Recuperación de información

29.3.1. Ejecución de consultas

29.3.2. Selección de registros

29.3.3. Uso de parámetros

29.4. Manipulación de la información

29.4.1. Altas, bajas y modificaciones

30 **PROGRAMACIÓN DE INTERFACES GRÁFICAS DE USUARIO** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5f #ce5g #ce5h #ev3 #opcional #ra1 #ra3 #ra5 #ra6* (est: 2020-05-13)

30.1. JFC y Swing

30.2. Componentes de Swing

30.3. Contenedores de nivel superior

30.3.1. JFrame

30.3.2. JDialog

30.3.3. JApplet

30.4. JComponent

30.5. Componentes de texto

30.5.1. JTextComponent

30.6. Marcos

30.7. Etiquetas

30.8. Botones

30.9. Arquitectura de modelos de Swing

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

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
19. Clases y objetos básicos en Java	#ce1a #ce1b #ce1c	#ce2e	#ce3f #ce3g	#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h					
20. Programación modular II	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4i #ce4j					
21. Programación genérica	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6f			
22. Estructuras de datos lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a			
23. Ordenación y búsqueda	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
24. Estructuras de datos no lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
25. Control de excepciones	#ce1a #ce1b #ce1c		#ce3d #ce3f #ce3g						
26. Java Collections Framework	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a #ce6b #ce6c #ce6d #ce6e #ce6f			
27. Principios y patrones de diseño	#ce1a #ce1b #ce1c		#ce3f #ce3g						
28. Calidad II	#ce1a #ce1b #ce1c		#ce3f #ce3g						
29. 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
30. Programación de interfaces gráficas de usuario	#ce1a #ce1b #ce1c		#ce3f #ce3g		#ce5f #ce5g #ce5h	×			

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 básicos, 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 aseguramiento 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 auto-aprendizaje 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.

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.

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

8.3. Online

- Plataforma **Ágora**² 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).
 - Los alumnos disponen, de forma totalmente gratuita, del **GitHub Student Developer Pack**⁴, que les ofrece servicios y herramientas con descuentos de hasta el 100 % con respecto al precio de mercado.
- **Dokuwiki**⁵ como herramienta de documentación colaborativa.

8.4. Bibliografía

8.4.1. Principal

- Apuntes y documentación *online*⁶ suministrados por el profesor.
- Documentación de Python⁷.
- Documentación de Java SE⁸.

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 <http://agora.iesdonana.org>

³ <https://classroom.github.com>

⁴ <https://education.github.com/pack>

⁵ <http://wiki.iesdonana.org>

⁶ <https://pro.iesdonana.org>

⁷ <https://docs.python.org/3/>

⁸ <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».*⁹

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. Modificaciones en el marco del COVID-19

Como consecuencia de las medidas excepcionales aplicadas en función de las instrucciones [4] emitidas por la Consejería de Educación para dar respuesta a la crisis sanitaria ocasionada por el COVID-19, se realizan las siguientes modificaciones en la presente programación.

12.1. Resultados de aprendizaje y criterios de evaluación

Los resultados de aprendizaje y sus criterios de evaluación asociados quedan estructurados de la siguiente manera:

- Resultados de aprendizaje y criterios de evaluación ya trabajados durante los dos primeros trimestres.
- Resultados de aprendizaje y criterios de evaluación a trabajar durante el tercer trimestre por el alumnado que estuviera siguiendo con regularidad la actividad docente no presencial.
- Resultados de aprendizaje y criterios de evaluación que no podrán trabajarse durante el presente curso y que, por consiguiente, deberán incorporarse al curso siguiente.

⁹ <https://twitter.com/RAEinforma/status/1111565711653113856>

12.1.1. Resultados de aprendizaje y criterios de evaluación ya trabajados durante los dos primeros trimestres

[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.

[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.

[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.

12.1.2. Resultados de aprendizaje y criterios de evaluación a trabajar durante el tercer trimestre

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

Criterios de evaluación:

CE4.i) Se han definido y utilizado interfaces.

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

[RA7] Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

Criterios de evaluación:

CE7.i) Se han escrito programas que utilicen arrays

CE7.j) Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.

CE7.k) Se han utilizado listas para almacenar y procesar información.

CE7.l) Se han utilizado iteradores para recorrer los elementos de las listas.

CE7.m) Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles.

CE7.n) Se han creado clases y métodos genéricos.

12.1.3. Resultados de aprendizaje y criterios de evaluación que no podrán trabajarse durante el presente curso

[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.

12.2. Contenidos y temporalización

La estructura de contenidos se adapta priorizándose los más básicos y fundamentales asociados a los resultados de aprendizaje que se han podido trabajar hasta el momento, y se define en función de los resultados de aprendizaje con los que están asociados.

En consecuencia, los contenidos se reparten de la siguiente manera:

- Contenidos ya trabajados durante los dos primeros trimestres.
- Contenidos a trabajar durante el tercer trimestre por el alumnado que estuviera siguiendo con regularidad la actividad docente no presencial.
- Contenidos que no podrán trabajarse durante el presente curso y que, por consiguiente, deberán incorporarse al curso siguiente.

12.2.1. Contenidos ya trabajados durante los dos primeros trimestres

Unidad didáctica
1. Introducción
2. Programación funcional I
3. Programación funcional II
4. Programación imperativa
5. Programación estructurada
6. Tipos de datos estructurados
8. Programación modular I
9. Abstracción de datos
13. Programación orientada a objetos
14. Relaciones entre clases Python
15. Introducción a la tecnología Java
16. Elementos básicos del lenguaje Java
17. Programación orientada a objetos en Java
18. Relaciones entre clases Java
19. Clases y objetos básicos en Java
20. Programación modular II

12.2.2. Contenidos a trabajar durante el tercer trimestre

Unidad didáctica
21. Programación genérica
26. Java Collections Framework

12.2.3. Contenidos que no podrán trabajarse durante el presente curso

Unidad didáctica
10. Entrada y salida de información
11. Calidad I
22. Estructuras de datos lineales
23. Ordenación y búsqueda
24. Estructuras de datos no lineales
25. Control de excepciones

Unidad didáctica
28. Calidad II
29. Gestión de bases de datos relacionales

Se considera que los contenidos anteriores, aunque no se hayan trabajado durante el presente curso, no resulta necesario trasladarlos a módulos profesionales de segundo curso, ya que en el módulo profesional «Desarrollo web en entorno servidor» de segundo curso se trabajan con suficiente profundidad todos los contenidos mencionados, pudiéndose alcanzar sin dificultad los resultados de aprendizaje asociados.

12.3. Metodología

Como consecuencia de la imposibilidad de asistencia presencial a clase, el desarrollo del resto del curso pasa a realizarse íntegramente a distancia a través de Internet. El modo de trabajo se basará en la publicación, a través de la plataforma Ágora, del material necesario que el alumno tendrá que trabajar, incluyendo las diferentes actividades a realizar. Se promoverá también la realización de audioconferencias intercaladas en el tiempo, con el objeto de resolver dudas y trabajar en directo los puntos de mayor dificultad, para lo cual se usarán herramientas como Discord, Google Meet o similar. El resto del tiempo se podrán resolver dudas y comunicarse con el profesor mediante correo electrónico corporativo, plataforma Ágora, Discord o Google Meet.

12.4. Actividades

Durante el tercer trimestre se realizarán dos tipos de actividades:

- **Actividades de refuerzo/recuperación:** para el alumnado que hubiera presentado dificultades de aprendizaje durante los dos primeros trimestres, así como para el alumnado que, no presentando tales dificultades, desee mejorar su calificación final.
- **Actividades de continuidad:** para el alumnado que estuviera siguiendo con regularidad la actividad docente no presencial, para el desarrollo de contenidos y competencias previstos para el tercer trimestre.

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.

12.5. Evaluación

12.5.1. Recuperación de los dos primeros trimestres

Los alumnos pendientes de evaluación positiva en alguna de las dos primeras evaluaciones deberán realizar las **actividades de refuerzo/recuperación** relativas a los trimestres correspondientes a las evaluaciones no superadas. Para determinar si una evaluación ha sido superada se comprobará la calificación obtenida por el alumno en dicha evaluación según se establece en el apartado de **calificaciones**.

12.5.2. Evaluación del tercer trimestre

La evaluación del tercer trimestre se llevará a cabo mediante las **actividades de continuidad** asociadas al mismo. Dicha evaluación será continua y tendrá carácter formativo y diagnóstico para valorar posibles desfases curriculares y los avances realizados, con el objeto de aportar valor añadido a los resultados obtenidos por el alumno en los dos primeros trimestres y planificar las medidas que sean necesarias adoptar para el próximo curso.

12.5.3. Evaluación final

En la evaluación y calificación final se tendrán fundamentalmente en consideración los resultados de los dos primeros trimestres y, a partir de ellos, se valorarán las actividades y pruebas realizadas por el alumnado en el tercer trimestre únicamente si benefician al alumno en el cálculo de la nota final.

12.6. Calificación

12.6.1. Calificaciones parciales del primer y segundo trimestres

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

Algoritmo 3 (Cálculo de la calificación parcial en el 1.º y 2.º trimestres)

```

if EV > 4.5:           # alumno aprobado que quiere subir nota
    NOTA = max(ACT, EV)
else:                 # alumno que tiene que recuperar el trimestre
    NOTA = min(ACT, EV)
  
```

Donde:

ACT: Media aritmética de las calificaciones de las **actividades de refuerzo/recuperación** del trimestre, valoradas del 0 al 10.

EV: Calificación obtenida con anterioridad en la evaluación parcial del trimestre correspondiente.

La evaluación parcial de cada trimestre se considerará aprobada si **NOTA** \geq 4,5.

12.6.2. Calificación parcial del tercer trimestre

La **calificación de la evaluación parcial del tercer trimestre** se calculará de la siguiente forma:

Algoritmo 4 (Cálculo de la calificación parcial del 3.º trimestre)

NOTA = **ACT**

Donde:

ACT: Media aritmética de las calificaciones de las **actividades de continuidad** del tercer trimestre, valoradas del 0 al 10.

12.6.3. Calificación final

La **calificación final** del módulo se calculará de la siguiente forma:

Algoritmo 5 (Cálculo de la calificación final)

```
media123 = EV1 * 0.33 + EV2 * 0.33 + EV3 * 0.33
media12 = EV1 * 0.5 + EV2 * 0.5
min123 = min(EV1, EV2, EV3)
min12 = min(EV1, EV2)

if media123 > media12:      # si la 3ª ev. favorece al alumno:
    if min123 ≥ 4:          # si tiene un 4 mínimo 4 en las tres ev.:
        NOTA = media123    # la nota es la media de las tres ev.
    else:                  # si no tiene un 4 mínimo en las tres ev.:
        NOTA = min123      # la nota es la mínima de las tres ev.
else:                      # si la 3ª ev. no favorece al alumno:
    if min12 ≥ 4:          # si tiene un 4 mínimo en las ev. 1 y 2:
        NOTA = media12    # la nota es la media de las ev. 1 y 2
    else:                  # si no tiene un 4 mínimo en las ev. 1 y 2:
        NOTA = min12      # la nota es la mínima de las ev. 1 y 2
```

Donde:

EV₁: Calificación de la primera evaluación, calculada según se establece en apartados anteriores sobre calificaciones parciales.

EV₂: Calificación de la segunda evaluación, calculada según se establece en apartados anteriores sobre calificaciones parciales.

EV₃: Calificación de la tercera evaluación, calculada según se establece en apartados anteriores sobre calificaciones parciales.

El módulo se considerará aprobado si **NOTA** \geq 4,5.

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)¹⁰.
- [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*.¹¹
- [4] *Instrucciones de 23 de abril de 2020*, de la Viceconsejería de Educación y Deporte, relativas a las medidas educativas a adoptar en el tercer trimestre del curso 2019/2020.

¹⁰ <http://www.juntadeandalucia.es/boja/boletines/2011/149/d/updf/d23.pdf\T1\textbackslash#page=17>

¹¹ http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/Instruccion_Pantallas/Instruccion_basica.pdf