



UTPL
La Universidad Católica de Loja

Modalidad Abierta y a Distancia

Programación Orientada a Objetos

Guía didáctica



Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Facultad de Ingenierías y Arquitectura

Departamento de Ciencias de la Computación y Electrónica

Programación Orientada a Objetos

Guía didáctica

Carrera	PAO Nivel
▪ <i>Tecnologías de la Información</i>	III

Autor:

Elizalde Solano René Rolando



Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Universidad Técnica Particular de Loja

Programación Orientada a Objetos

Guía didáctica

Elizalde Solano René Rolando

Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojacialtda@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-25-621-8



**Reconocimiento-NoComercial-CompartirIgual
4.0 Internacional (CC BY-NC-SA 4.0)**

Usted acepta y acuerda estar obligado por los términos y condiciones de esta Licencia, por lo que, si existe el incumplimiento de algunas de estas condiciones, no se autoriza el uso de ningún contenido.

Los contenidos de este trabajo están sujetos a una licencia internacional Creative Commons **Reconocimiento-NoComercial-CompartirIgual 4.0 (CC BY-NC-SA 4.0)**. Usted es libre de **Compartir** – copiar y redistribuir el material en cualquier medio o formato. **Adaptar** – remezclar, transformar y construir a partir del material citando la fuente, bajo los siguientes términos: **Reconocimiento** – debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatario. **No Comercial** – no puede hacer uso del material con propósitos comerciales. **Compartir igual** – Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original. No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia. <https://creativecommons.org/licenses/by-nc-sa/4.0/>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Índice

1. Datos de información.....	8
1.1. Presentación - Orientaciones de la asignatura	8
1.2. Competencias genéricas de la UTPL.....	8
1.3. Competencias específicas de la carrera.....	8
1.4. Problemática que aborda la asignatura en el marco del proyecto.....	9
1.5. Competencias específicas de la asignatura.....	9
2. Metodología de aprendizaje.....	10
3. Orientaciones didácticas por resultados de aprendizaje.....	11
Primer bimestre	11
Resultados de aprendizaje 1 y 2.....	11
Contenidos, recursos y actividades de aprendizaje.....	11
Semana 1	12
 Unidad 1. Introducción a Programación Orientada a Objetos.....	12
1.1. El paradigma de Programación Orientada a Objetos.....	12
1.2. Objetos y clases	14
1.3. Uso métodos y encapsulación	18
 Semana 2	22
1.4. Diagramas de clases en Programación Orientada a Objetos.....	22
1.5. Uso de la arquitectura modelo-vista-controlador.....	24
Actividad de aprendizaje recomendada.....	27
Autoevaluación 1.....	28
Resultados de aprendizaje 3 y 4.....	32
Contenidos, recursos y actividades de aprendizaje.....	32
 Semana 3	33

Unidad 2. Estructura y creación de algoritmos orientados a objetos	33
2.1. Creación de miniespecificaciones orientados a objetos	33
2.2. Uso de constructores.....	41
Resultados de aprendizaje 1 y 2.....	44
Contenidos, recursos y actividades de aprendizaje	44
Semana 4	44
2.3. Uso de estructuras de selección simple y compuesta en POO	45
Actividades de aprendizaje recomendadas.....	49
Autoevaluación 2.....	51
Resultados de aprendizaje 1 y 3.....	57
Contenidos, recursos y actividades de aprendizaje.....	57
Semana 5	57
Unidad 3. Estructuras de repetición en Programación Orientada a Objetos	58
3.1. Uso de estructura de repetición do-while en POO	58
3.2. Uso de Estructura de repetición while en POO	60
Semana 6	64
3.3. Uso de Estructura de repetición for en POO	64
Actividades de aprendizaje recomendadas.....	67
Autoevaluación 3.....	68
Actividades finales del bimestre.....	78
Semana 7	78
Actividades de aprendizaje recomendadas.....	78
Semana 8	80
Segundo bimestre	82

<p>Resultados de aprendizaje 1, 2 y 5.....</p> <p>Contenidos, recursos y actividades de aprendizaje.....</p> <p>Semana 9</p> <p>Unidad 4. Estructuras de datos en Programación Orientada a Objetos</p> <p> 4.1. Uso de arreglos de tipos de datos simples o primitivos......</p> <p> 4.2. Uso de arreglos de tipos de datos de objetos</p> <p>Semana 10</p> <p> 4.3. Ejercicios propuestos usando arreglos</p> <p>Actividades de aprendizaje recomendadas.....</p> <p>Autoevaluación 4.....</p> <p>Resultados de aprendizaje 1, 3 ,7 y 8.....</p> <p>Contenidos, recursos y actividades de aprendizaje.....</p> <p>Semana 11</p> <p>Unidad 5. Herencia en Programación Orientada a Objetos</p> <p> 5.1. Concepto de herencia</p> <p> 5.2. Uso de diagramas de clase con herencia.....</p> <p>Semana 12</p> <p> 5.3. Algoritmos orientados a objetos con herencia.....</p> <p>Actividad de aprendizaje recomendada.....</p> <p>Autoevaluación 5.....</p> <p>Resultados de aprendizaje 6, 7, 8 y 9.....</p> <p>Contenidos, recursos y actividades de aprendizaje.....</p> <p>Semana 13</p> <p>Unidad 6. Polimorfismo en Programación Orientada a Objetos</p> <p> 6.1. Concepto de polimorfismo</p> <p> 6.2. Uso de diagramas de clase con polimorfismo.....</p>	<p>Índice</p> <p>Primer bimestre</p> <p>Segundo bimestre</p> <p>Solucionario</p> <p>Referencias bibliográficas</p>
--	---

Semana 14	134
6.3. Algoritmos orientados a objetos con polimorfismo.....	134
Actividad de aprendizaje recomendada.....	138
Autoevaluación 6.....	139
Actividades finales del bimestre.....	147
Semana 15	147
Actividad de aprendizaje recomendada.....	147
Semana 16	149
4. Solucionario	151
5. Referencias bibliográficas	163

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



1. Datos de información

1.1. Presentación - Orientaciones de la asignatura



1.2. Competencias genéricas de la UTPL

- Trabajo en equipo.

1.3. Competencias específicas de la carrera

- Construir modelos específicos de ciencias de la computación mediante esquemas matemáticos y estadísticos, para propiciar el uso y explotación eficiente de datos e información.

- Implementar aplicaciones de baja, mediana y alta complejidad integrando diferentes herramientas y plataformas para dar solución a requerimientos de la organización.

1.4. Problemática que aborda la asignatura en el marco del proyecto

El proyecto que permitirá poner en práctica los resultados de aprendizaje alcanzados en el desarrollo de las actividades académicas de las diferentes asignaturas del ciclo, tiene relación con la siguiente temática: creación de aplicaciones computacionales sustentadas en modelos matemáticos aplicados a un contexto social.

1.5. Competencias específicas de la asignatura

- Construir algoritmos para resolver problemas reales mediante el uso del enfoque orientado a objetos.
- Programar algoritmos para resolver problemas usando orientación de objetos y lenguajes de alto nivel.
- Abstraeer problemas del mundo real identificando las características esenciales de sus componentes y modela su comportamiento en el contexto de operación del computador.



2. Metodología de aprendizaje

La metodología de aprendizaje que se utiliza se denomina *Blended Learning*.

A través de esta metodología, el proceso de enseñanza aprendizaje se divide en:

- Trabajo autónomo.
- Actividades síncronas y asíncronas.

Lo que le permite al estudiante organizar su tiempo para cumplir con las tareas propuestas en cada una de las unidades de estudio de la asignatura. Además, permite que el docente acompañe al estudiante a través de chat de tutoría permanente para dar solución y aclarar dudas; finalmente el profesional en formación recibe un trato personalizado que le ayuda a avanzar en la adquisición de las competencias de esta asignatura.

Estimado estudiante, puede revisar información relacionada a la metodología planteada en el siguiente enlace [<https://aretio.hypotheses.org/2437>].

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultados de aprendizaje 1 y 2

- Diseña, implementa, prueba y depura programas sencillos en un lenguaje de programación orientado a objetos.
- Describe cómo el mecanismo de la clase soporta la encapsulación y la ocultación de la información.

Contenidos, recursos y actividades de aprendizaje

El estudio del paradigma de programación de orientación a objetos debe iniciar con el abordaje de los conceptos primarios del enfoque, mismos que permitan consolidar un conjunto de conocimientos sólidos para su posterior uso en la resolución de problemas; por tal

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

razón en la primera y segunda semana de actividades académicas se quiere alcanzar los resultados de aprendizaje.



Semana 1



Unidad 1. Introducción a Programación Orientada a Objetos

Apreciado estudiante: iniciemos nuestro trabajo académico haciendo una revisión de la presente guía de estudio que tiene como eje central el estudio a nivel conceptual y práctico del paradigma de Programación Orientada a Objetos POO u *Object Oriented Programming* (OPP) en inglés. Se inicia con una introducción al nuevo paradigma de programación; además se indican pautas importantes para describir e identificar conceptos importantes como son los objetos y clases; finalmente es indispensable entender la creación y uso de métodos como medio a través del cual se pueden realizar acciones para un conjunto de objetos.

1.1. El paradigma de Programación Orientada a Objetos

Para abordar y estudiar el paradigma de programación se solicita revisar los siguientes recursos.

- Texto básico en el capítulo “Introducción a la programación”, sección Evolución de los paradigmas de programación.
- Guía didáctica en la unidad *Introducción a Programación Orientada a Objetos*, sección *El Paradigma de Programación Orientada a Objetos*.

En los recursos indicados se encuentra explicado las características del paradigma de Orientación a Objetos. En la figura 1 se señalan algunas ideas en torno al paradigma mencionado.

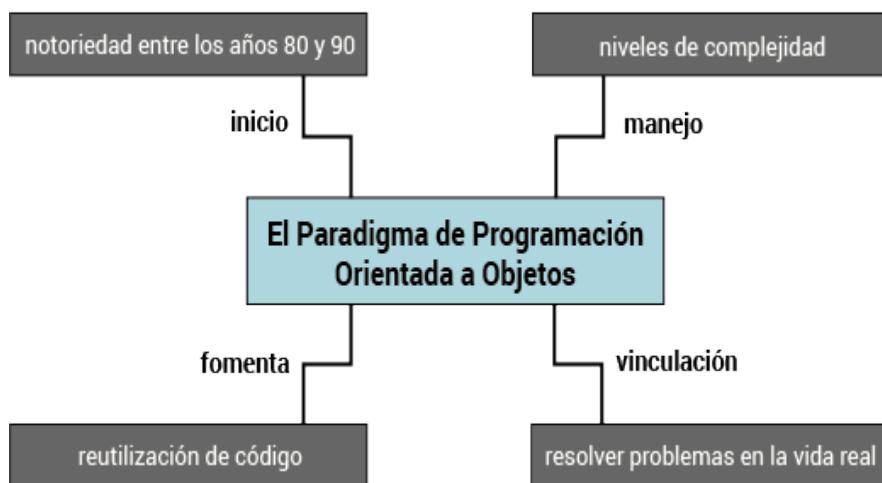


Figura 1. Paradigma de Programación Orientada a Objetos.

Elaborado por: Elizalde, R. (2019).

En el desarrollo de las subsiguientes temáticas usted estudiará conceptos e ideas que permiten entender cómo generar soluciones informáticas basadas en Programación Orientada a Objetos.

1.2. Objetos y clases

1.2.1. Objetos

Para el estudio del presente apartado, le invito a revisar el contenido de:

- *Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Objetos.*
- *Guía didáctica en la unidad “Introducción a Programación Orientada a Objetos”, sección Objetos y Clases, subsección Objetos.*

Las características de la Programación Orientada a Objetos se resumen en la figura 2.

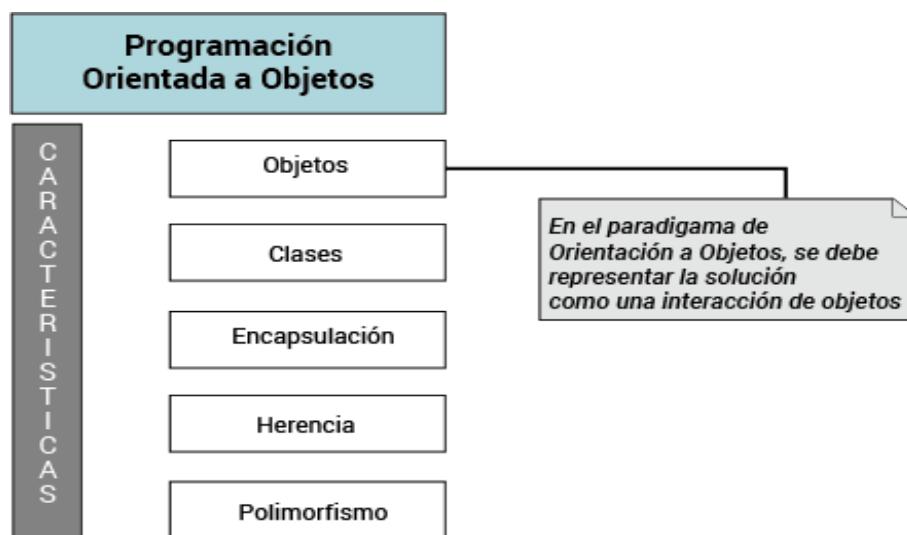


Figura 2. Características de la Programación Orientada a Objetos.

Elaborado por: Elizalde, R. (2019).

En la figura 3 se hace referencia a la interacción de objetos como base para solucionar las problemáticas en el Paradigma de Orientación a Objetos.

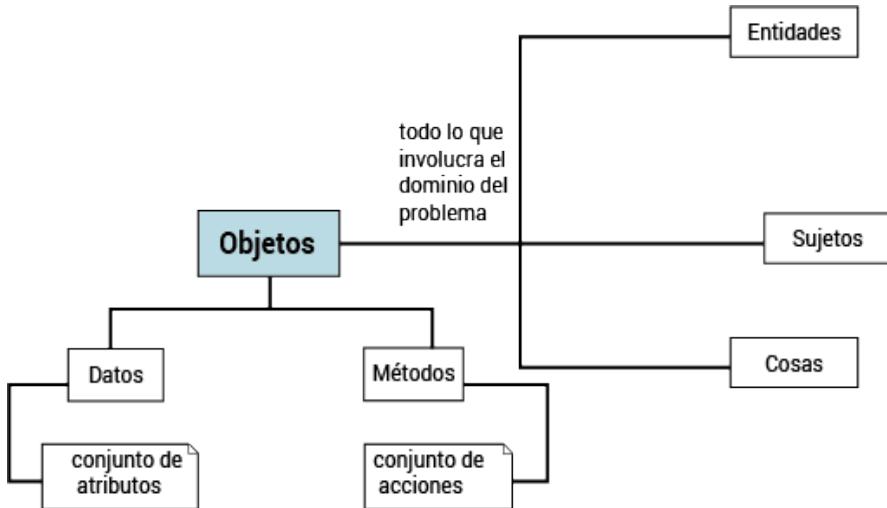


Figura 3. Objetos formados por datos y métodos.

Elaborado por: Elizalde, R. (2019)

En las problemáticas los objetos son las entidades, sujetos o cosas involucradas y que se deben tomar en consideración para formular una solución y están formados por datos y métodos; en la figura 3 se describe lo mencionado.

Importante: estimado estudiante, en otros paradigmas de programación los métodos tienen otras denominaciones como procedimientos, módulos, funciones, subrutinas, entre otros.

Para la representación gráfica, mediante un diagrama se toma en consideración algunas pautas, las mismas que se señalan en la figura 4.

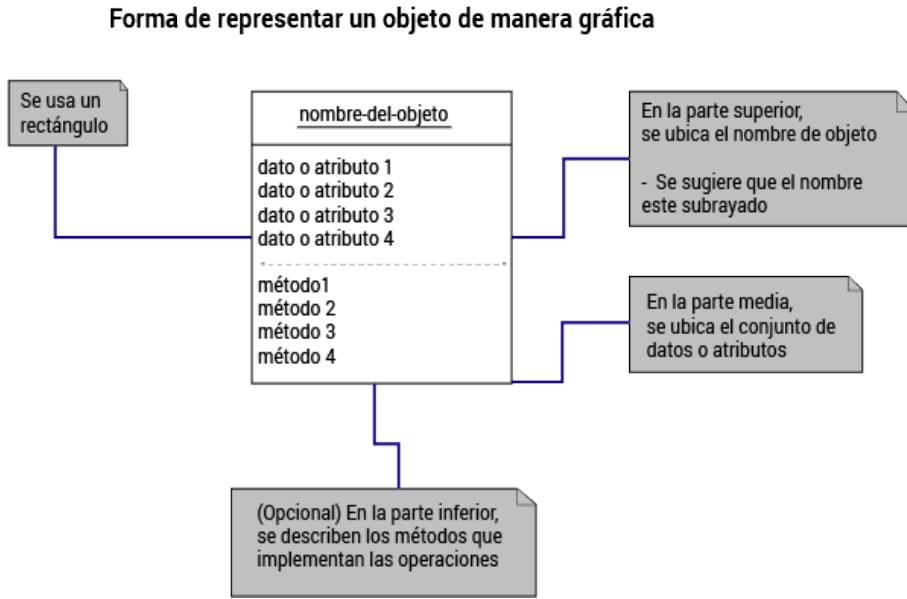


Figura 4. Forma de representar un objeto de manera gráfica.

Elaborado por: Elizalde, R. (2019)

Estimado estudiante, para finalizar esta parte de estudio, se solicita revisar el Ejemplo 1 de la sección indicada de la guía didáctica.

1.2.2. Clases

Para el estudio del presente apartado, le invito a revisar el contenido de:

- Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Clases y su relación con los objetos.
- Guía didáctica en la unidad “Introducción a Programación Orientada a Objetos”, sección Objetos y Clases, subsección Clases.

Una clase es considerada como la representación abstracta que agrupa un conjunto de objetos que tienen en común: una misma estructura (datos) y un mismo comportamiento (métodos); es decir la clase se convierte en una plantilla que permite crear objetos.

La representación de una clase a través de un diagrama, lo describimos en la figura 5.

Representación gráfica de una clase



Figura 5. Representación de una clase en diagrama.
Elaborado por: Elizalde, R. (2019)

Representación de una clase en diagrama

A continuación, en la figura 6, se describe la representación de una clase y sus instancias (objetos).

Representación en diagrama de la relación entre una clase y sus instancias.

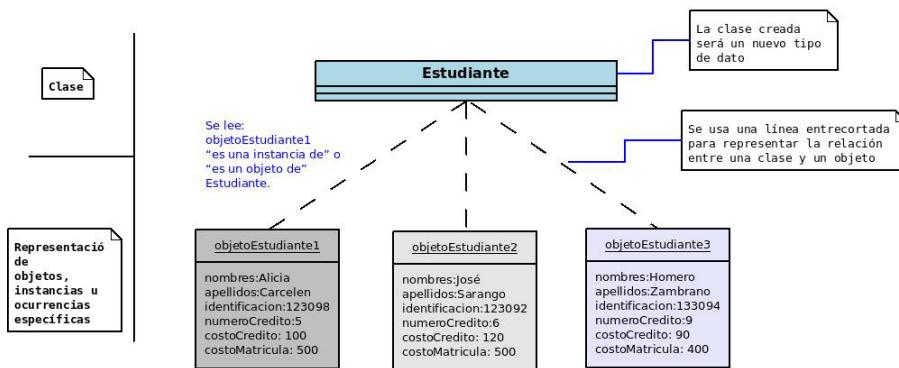


Figura 6. Representación en diagrama de la relación de una clase y sus instancias.

Elaborado por: Elizalde, R. (2019)

Estimado estudiante, a través del siguiente enlace, puede descargar los archivos del ejemplo descrito en la guía didáctica en la subsección Clases.

Diagrama

Representación de una clase y sus instancias

1.3. Uso métodos y encapsulación

Estimado estudiante, para una mejor comprensión de las temáticas del presente apartado, se solicita revisar el contenido del texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Métodos y encapsulación.

1.3.1. Métodos

Para el estudio del presente apartado, le invito a revisar el contenido de:

- *Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Métodos.*
- *Guía didáctica en la unidad “Introducción a programación orientada a objetos”, sección Uso métodos y encapsulación, subsección Métodos.*

En la figura 7 se señalan algunas características y la finalidad del uso de los métodos.

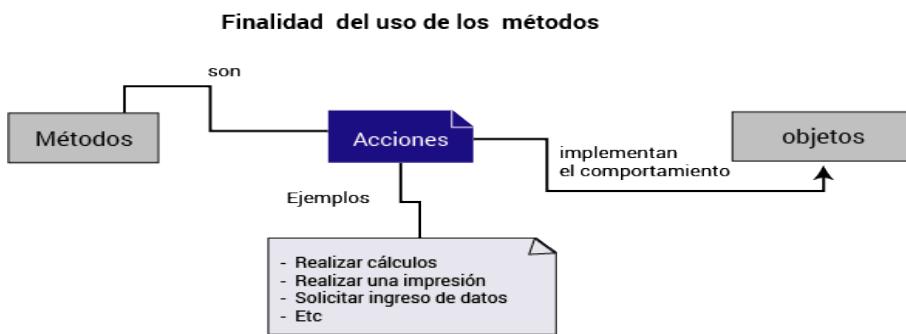


Figura 7. Finalidad del uso de los métodos.

Elaborado por: Elizalde, R. (2019)

1.3.2. Encapsulación

Para el estudio del apartado, previamente le invito a revisar el contenido de:

- *Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Métodos.*
- *Guía didáctica en la unidad “Introducción a programación orientada a objetos”, sección Uso métodos y encapsulación, subsección Encapsulación.*

En relación con los contenidos de los recursos antes indicados, se puede destacar la siguiente información referenciada en la figura 8.

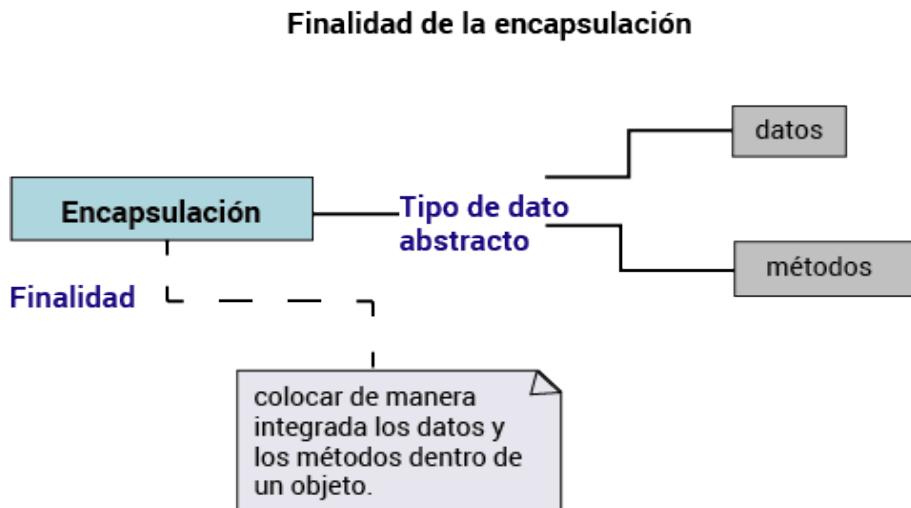


Figura 8. Finalidad de la encapsulación.

Elaborado por: Elizalde, R. (2019)

Recursos de aprendizaje

- Lea los contenidos y ejemplos de la unidad 1. Introducción a Programación Orientada a Objetos.

LECTURA: Elizalde, R (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 1 de la Guía, brindan a los estudiantes un compendio y exemplificación relacionada con las temáticas sobre introducción a la Programación Orientada a Objetos, clases, objetos, métodos y encapsulación. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 8 Programación Orientada a Objetos usando el diagrama de clases; páginas 222-235.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado con objetos en el ámbito del Paradigma de Orientación a Objetos con la finalidad de comprender el qué, cómo y cuándo utilizar y reconocer los objetos; adicionalmente, se encuentra información sobre el manejo y representación de instancias de clases; el uso correcto de los métodos y la importancia de la encapsulación se aborda en la lectura. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre una introducción a Programación Orientada a Objetos

VIDEO:

Escuela de Informáticos. (2016). *Programación Orientada a Objetos (POO) - Objetos y Clases*. [Archivo de video]. Recuperado de [Programacion Orientada a Objetos \(POO\) - Objetos y Clases](#)

En el video se puede reforzar los conocimientos referentes a la Programación Orientada a Objetos; en el mismo se explican preguntas como: ¿Qué es un objeto?; ejemplos de objetos; ¿Qué es una clase? y ejemplos de clases.



Semana 2

1.4. Diagramas de clases en Programación Orientada a Objetos

Para el estudio del presente apartado, se solicita revisar el contenido de:

- Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases”, sección Diseño del diagrama de clases.
- Guía didáctica en la unidad “Introducción a Programación Orientada a Objetos”, sección Diagramas de Clases en Programación Orientada a Objetos.

A continuación, se indican las pautas para representar una clase mediante un diagrama, pero con la inclusión de sus datos y métodos; a través de la figura 9. Se hace referencia a la visibilidad que puedan tener los datos y métodos; este tema lo abordaremos en la próxima”, sección con más profundidad.

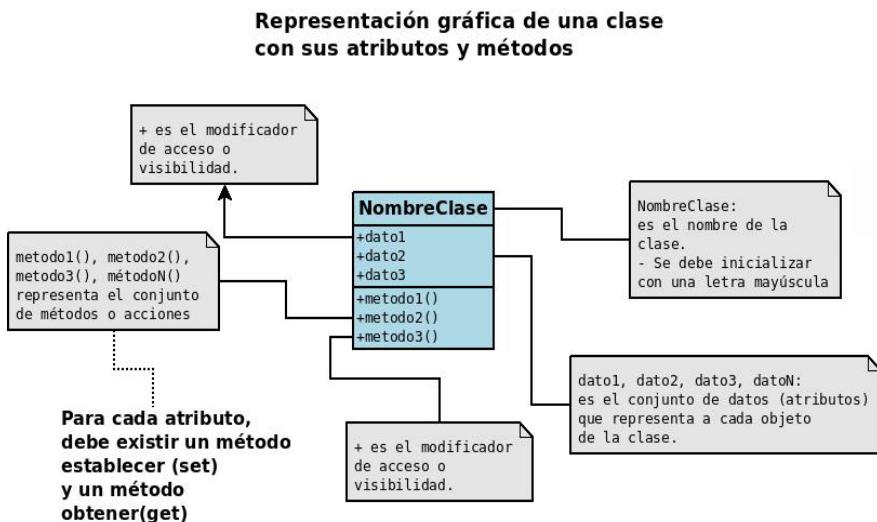


Figura 9. Representación gráfica de una clase con sus atributos y métodos.
Elaborado por: Elizalde, R. (2019)

1.4.1. Modificadores de acceso

Cuando se habla de datos o métodos se debe tener en consideración los modificadores de acceso o visibilidad que se puede asociar a los datos (atributos) y métodos; el tipo de modificador de acceso se usa tanto para la generación de diagramas de clase, miniespecificaciones y soluciones en lenguajes de alto nivel.

Se solicita revisar las tablas 1, 2 y 3 del apartado Modificadores de Acceso de la guía didáctica; con el objetivo de identificar el modificador de acceso, el símbolo como se lo representa y usos que permite cada uno de ellos.

Estimado estudiante, a través del siguiente enlace usted puede descargar los archivos del ejemplo de representación mediante el diagrama de una clase con atributos y métodos; relacionado con el ejemplo descrito en la subsección Modificadores de acceso de la guía didáctica.

Diagrama

Representación de diagrama de clase con atributos y métodos

1.5. Uso de la arquitectura modelo-vista-controlador

Para el estudio del presente apartado, se solicita revisar el contenido de:

- *Texto básico en el capítulo “Programación orientada a objetos usando el diagrama de clases , sección Arquitectura modelo-vista-controlador.*
- *Guía didáctica en la unidad “Introducción a Programación Orientada a Objetos”, sección Uso de la arquitectura modelo-vista-controlador.*

Luego de revisar los contenidos anteriores, hemos generado la figura 10 relacionada con la arquitectura del modelo-vista-controlador.

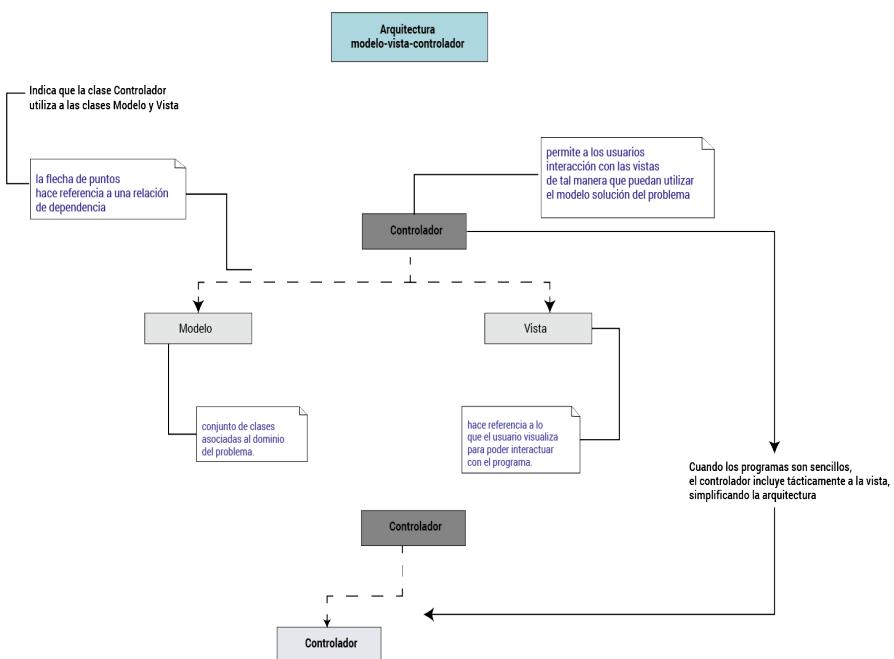


Figura 10. Arquitectura modelo - vista – controlador.

Elaborado por: Elizalde, R. (2019)

Al culminar nuestra primera unidad, es momento de verificar nuestros conocimientos con la realización y desarrollo de la autoevaluación.

Listo para iniciar. ¡Adelante!

Recurso de aprendizaje

- Lea los contenidos y ejemplos de la unidad 1. Introducción a Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

- Los contenidos de la unidad 1 de la guía, brindan a los estudiantes un compendio y exemplificación relacionados con las temáticas sobre la generación de diagramas de clases y arquitectura modelo-vista-controlador en el ámbito de la Programación Orientada a Objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.
- Revise los conceptos y ejercicios de la unidad 8 Programación Orientada a Objetos usando el diagrama de clases; páginas 222-235.

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. México. Alfaomega

La lectura permite al estudiante comprender lo relacionado al Diseño de Diagramas de Clases en cuanto a los modificadores de acceso y la generación de instancias y el uso de la arquitectura modelo-vista-controlador en la solución de problemas. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre Diagramas de Clases en Programación Orientada a Objetos.

VIDEO: ForrestKnight. (2016). *How To Create a UML Diagram Using DIA Diagram Editor*. [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=f-leQbc2o5k>

En el recurso presentado, se realiza una explicación detallada de cómo utilizar el programa DIA-UML para la creación de diagramas de clases. Entre las características presentadas se tiene: para cada clase se indica la forma de agregar nombre, atributos y operaciones; y cómo relacionar las clases.



Actividad de aprendizaje recomendada

Actividad 1

- **Actividad de aprendizaje**

La autoevaluación 1 se plantea a través de preguntas relacionadas con la Programación Orientada a Objetos usando diagramas de clase. Se solicita revisar la unidad 1 de la presente guía y el capítulo "Programación Orientada a Objetos usando el diagrama de clases" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección. Solucionario de la guía didáctica, en cuanto a la autoevaluación 1.



Autoevaluación 1

A continuación, se presentan preguntas relacionadas con la programación orientada a objetos usando diagramas de clase. Se solicita revisar la unidad 1 de la presente guía y el capítulo "Programación orientada a objetos usando el diagrama de clases" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. Para representar soluciones a través de la Programación Orientada a Objetos, ¿cómo se debe representar a la solución?
 - a. Manejo aislado de objetos.
 - b. Interactividad de objetos.
 - c. Sin manejo de objetos
2. ¿Cómo se identifican los objetos?
 - a. En una problemática los objetos son los verbos.
 - b. En una problemática los objetos son los sustantivos
 - c. En una problemática los objetos son los adverbios
3. ¿Cómo se conforman los objetos?
 - a. Se forman con datos (atributos) y métodos.
 - b. Se forman con clases y subclases.
 - c. Se forman con datos(atributos) protegidos y datos privados

Índice

Primer bimestre

Segundo bimestre

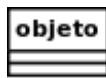
Solucionario

Referencias bibliográficas

4. ¿Cuáles son otras denominaciones que se le pueden dar a los métodos?

- a. Funciones, clases
- b. Procedimientos, funciones
- c. Objetos, funciones

5. Identifique la forma correcta de representar un objeto en diagrama.



Opción A



Opción B



Opción C

- a. Opción A.
- b. Opción B.
- c. Opción C.

6. Se requiere representar una colección de objetos que contenga los siguientes datos: nombres, apellidos, identificación, número de créditos. Identifique la colección correcta para lo solicitado.

Opción A	
objectIdEstudiante1	objectIdEstudiante2
nombres:Alicia apellidos:Carcenén identificacion:123098 numeroCredito:5	nombres:José apellidos:Andrade identificacion:128098 numeroCredito:10

Opción B

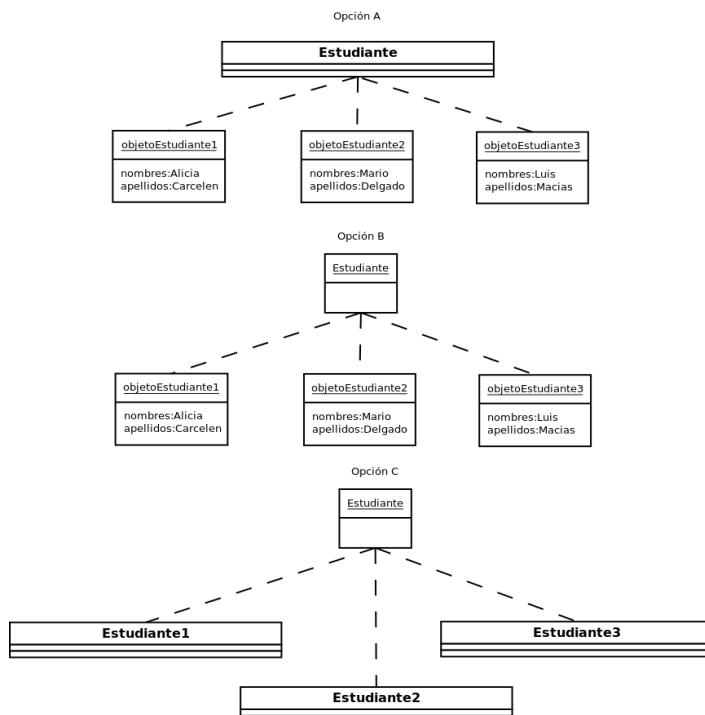
Opción B	
objectIdEstudiante1	objectIdEstudiante2
nombres:Alicia apellidos:Carcenén identificacion:123098	nombres:José apellidos:Andrade identificacion:128098 numeroCredito:10

Opción C

Opción C	
objectIdEstudiante1	objectIdEstudiante2
nombres:Alicia apellidos:Carcenén identificacion:123098 numeroCredito:5	nombres:Andrade apellidos:Andrade identificacion:128098 numeroCredito:10

- a. Opción A.
- b. Opción B.
- c. Opción C.

7. Identifique las características de una clase en Programación Orientada a Objetos.
- Conjunto de objetos que tienen los mismos datos y métodos.
 - Conjunto de objetos que tienen los mismos datos y diferentes métodos.
 - Conjunto de objetos que tienen diferentes datos y métodos.
8. Identifique la forma correcta de representar una clase y sus instancias, con base en las siguientes figuras.



- Opción A.
- Opción B.
- Opción C.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

9. Identifique el diagrama de una clase que contenga las siguientes características:

- Datos o Atributos: 1 atributo privado y 1 atributo protegido.
- Métodos: 1 método público, 2 métodos protegidos.

Opción A

NombreClase
-dato1
#dato2
#metodo1()
-metodo2()
-metodo3()

Opción B

NombreClase
-dato1
#dato2
+metodo1()
#metodo2()
#metodo3()

Opción C

NombreClase
+dato1
#dato2
+metodo1()
#metodo2()
#metodo3()

- a. Opción A.
- b. Opción B.
- c. Opción C.

10. Identifique las tres capas de la arquitectura que se usa en el desarrollo de soluciones bajo el paradigma orientado a objetos.

- a. Modelo-Vista-Objetos.
- b. Modelo-Vista-Controlador.
- c. Modelo-BaseDeDatos-Controlador.

Ir al solucionario

Resultados de aprendizaje 3 y 4

- Describe la relación entre la estructura estática de la clase y la estructura dinámica de las instancias de la clase.
- Describe cómo los constructores y destructores se relacionan con la vida de un objeto.

Contenidos, recursos y actividades de aprendizaje

Por lo indicado, es necesario entender y aprender los procedimientos para la creación de miniespecificaciones y diagramas de clases en el ámbito de la Programación Orientada a Objetos y su relación con conceptos estudiados en ciclos académicos anteriores como las estructuras de decisión, que permitan generar soluciones robustas a problemáticas planteadas de la vida cotidiana.

Los contenidos a desarrollar en la semana 3 son los relacionados a la unidad 2. Estructura y creación de algoritmos orientados a objetos, los subtemas específicos son: creación de miniespecificaciones orientadas a objetos y uso de constructores.

A nivel de miniespecificación se indican las formas correctas de creación de clases, atributos y métodos; además de la correcta creación de una miniespecificación completa con base en un diagrama de clase. El uso de constructores en la creación de clases se estudia como concepto importante en el proceso de entendimiento de Programación Orientada a Objetos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Semana 3



Unidad 2. Estructura y creación de algoritmos orientados a objetos

2.1. Creación de miniespecificaciones orientados a objetos

Para abordar la temática de creación de miniespecificaciones orientadas a objetos, es necesario revisar los siguientes contenidos:

- En el texto básico la unidad “Programación orientada a objetos aplicando la estructura de secuenciación, sección Diseño de algoritmos OO usando la secuenciación en seudocódigo.
- En la guía didáctica la unidad 2 “Estructura y creación de algoritmos orientados a objetos, sección Creación de miniespecificaciones orientados a objetos.

En la guía didáctica se hace referencia a los aspectos a seguir para la construcción y desarrollo de una miniespecificación; en la figura 11, detallamos algunas directrices para la creación de miniespecificaciones.

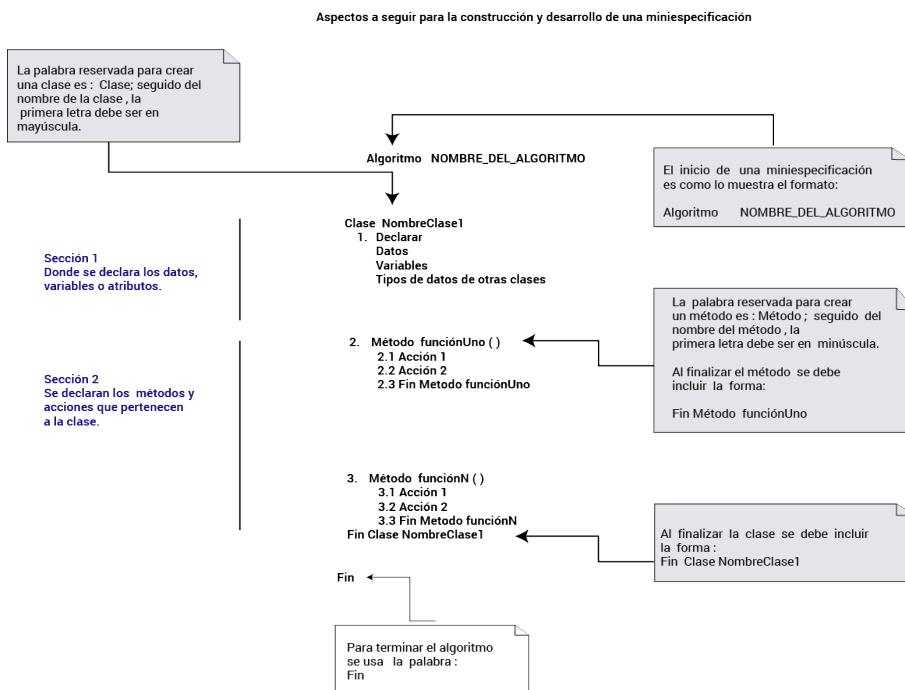


Figura 11. Aspectos a seguir para la construcción y desarrollo de una miniespecificación.

Elaborado por: Elizalde, R. (2019)

Estimado estudiante: en el guía didáctica en la sección “Creación de Miniespecificaciones Orientados a Objetos”, se describe un ejemplo que permite resolver la problemática: CALCULAR COSTO DE MATRICULA DE UN ESTUDIANTE.

En los siguientes enlaces, puede descargar los archivos del ejemplo mencionado. Se tiene la solución en diagrama de clases, la miniespecificación e implementación en lenguaje de programación Java.

Diagrama

Solución del ejemplo planteado en diagrama

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Miniespecificación

[Solución del ejemplo planteado en miniespecificación.](#)

Lenguaje de Programación

[Solución del ejemplo planteado en un lenguaje de programación.](#)

En las figuras 12 y 13 relacionamos las soluciones (miniespecificación e implementación en lenguaje de programación) descritas en los enlaces anteriores.

Explicación de ejemplo a través de diagrama de clases, miniespecificación y lenguaje de programación

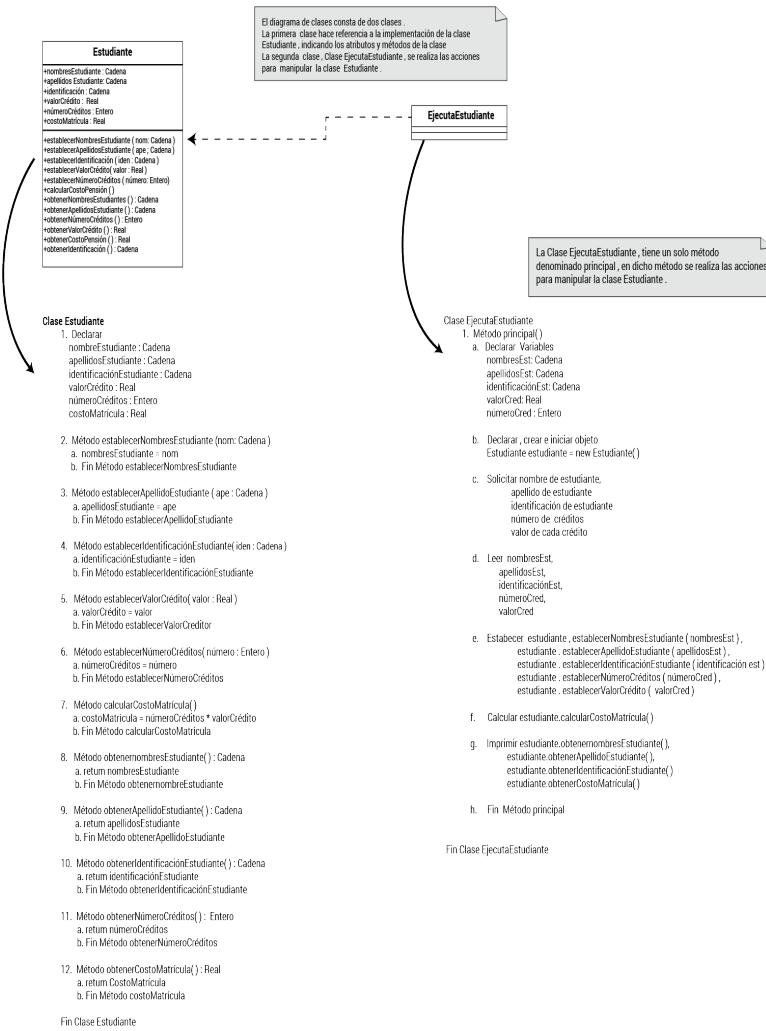


Figura 12. Explicación de ejemplo a través de diagrama de clases y miniespecificación.

Elaborado por: Elizalde, R. (2019)

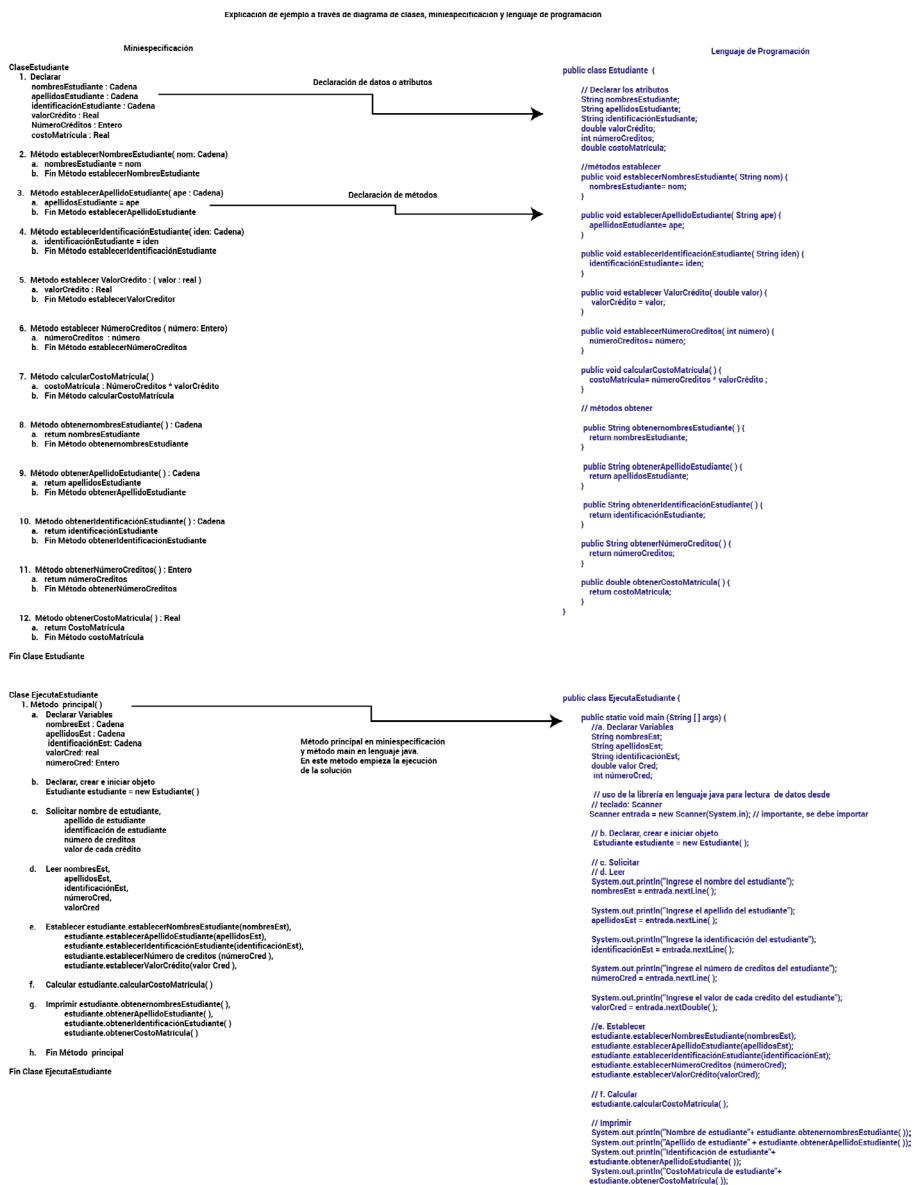


Figura 13. Explicación de ejemplo a través de miniespecificación y lenguaje de programación.

Elaborado por: Elizalde, R. (2019)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

El lenguaje de programación que se usa en el desarrollo de ejemplos y ejercicios es Java; en relación con lo mencionado, listamos los siguientes pasos que usted debe seguir para ejecutar un programa en lenguaje Java en su equipo personal.

Para poder ejecutar un programa en lenguaje Java usted debe realizar lo siguiente:

1. Tener instalado en lenguaje de programación Java ([proceso de instalación de lenguaje Java](#))
2. Tener instalado el IDE de programación Netbeans ([proceso de instalación del IDE Netbeans](#))
3. Ejemplificación a través del enlace ([Solución del ejemplo planteado en un lenguaje de programación](#)); descargar el archivo comprimido con extensión .zip (EjemploEstudiante.zip)
4. Luego de la descarga del archivo comprimido .zip; descomprimir en su computadora.
5. Abrir el IDE Netbeans > Archivo > Abrir Proyecto > ubicarse en la carpeta donde descomprimió el archivo y dar clic para abrir.

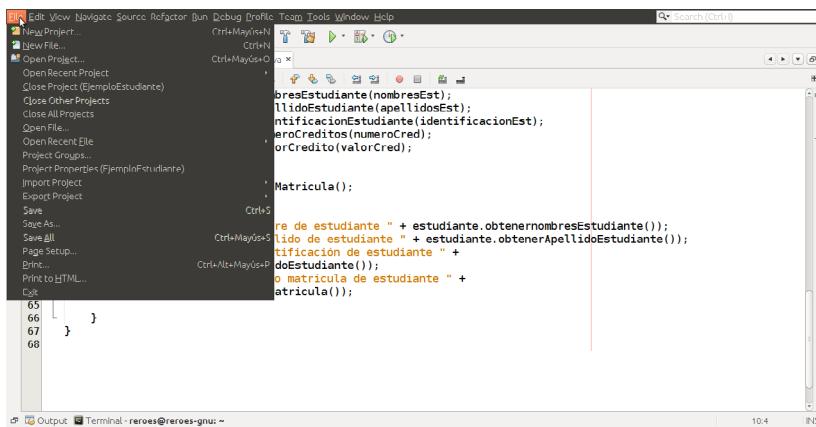


Figura 14. Abrir proyecto en Netbeans - paso 5.

Elaborado por: Elizalde, R. (2019)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

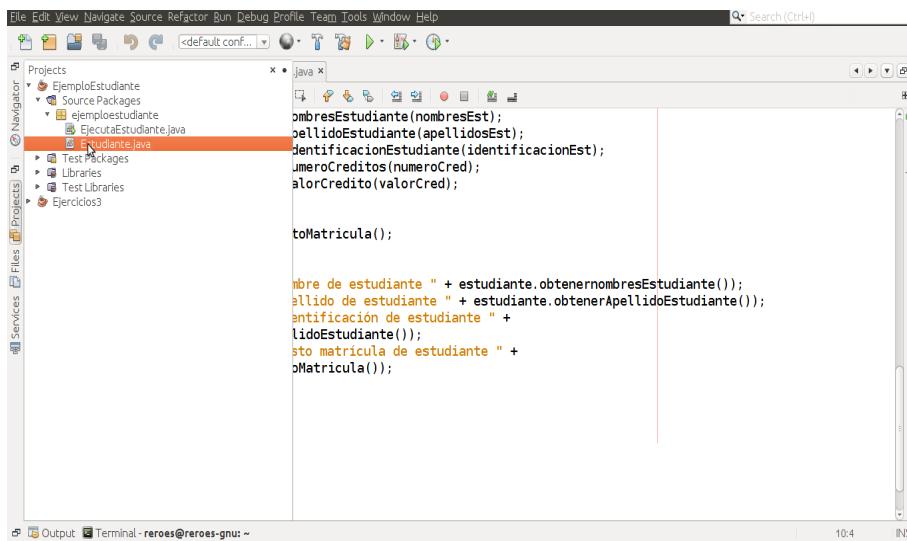


Figura 15. Abrir proyecto en Netbeans - paso 6.

Elaborado por: Elizalde, R. (2019)

6. Dar clic en la clase Estudiante para revisar el código.

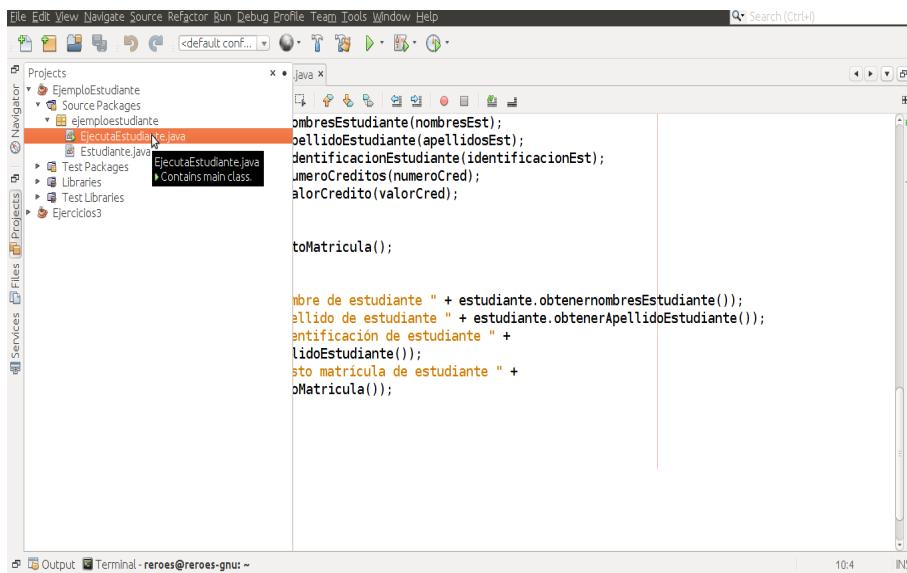


Figura 16. Abrir proyecto en Netbeans - paso 7.

Elaborado por: Elizalde, R. (2019)

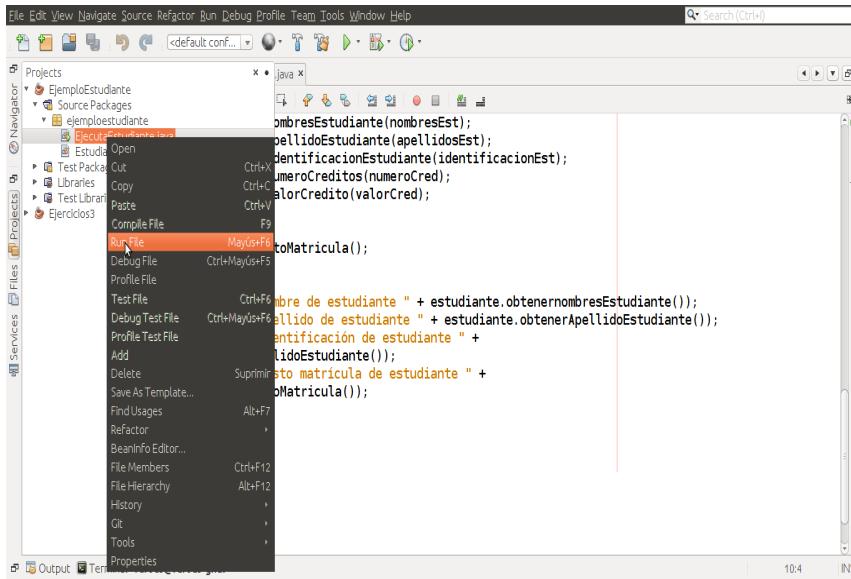


Figura 17. Abrir proyecto en Netbeans - paso 8.1.

Elaborado por: Elizalde, R. (2019)

9. Ingresar los valores que solicita el programa.
 10. Visualizar la salida generada por el programa.

```
run:  
Ingresar el nombre del estudiante  
René  
Ingresar el apellido del estudiante  
Elizalde  
Ingresar la identificación del estudiante  
110033  
Ingresar el número de créditos del estudiante  
8  
Ingresar el valor de cada crédito del estudiante  
100  
Nombre de estudiante René  
Apellido de estudiante Elizalde  
Identificación de estudiante 10033  
Costo matrícula de estudiante 800.0  
BUILD SUCCESSFUL (total time: 43 seconds)
```

Figura 18. Abrir proyecto en Netbeans - paso 10.

Elaborado por: Elizalde, R. (2019)

Para la ejecución de los próximos ejemplos descritos en la guía didáctica y en el presente documento que hagan relación a lenguaje Java, se deberá seguir los pasos anteriores.

2.2. Uso de constructores

Para abordar la temática sobre el uso de constructores, usted debe realizar una lectura de los siguientes recursos.

- *En el texto básico la unidad “Programación orientada a objetos aplicando la estructura de secuenciación , sección Constructores y destructores.*
- *En la guía didáctica la unidad 2. “Estructura y creación de algoritmos orientados a objetos , sección Uso de constructores.*

A continuación, en la figura 19 se describen algunas nociones importantes a considerar para la implementación de constructores en el desarrollo de soluciones.

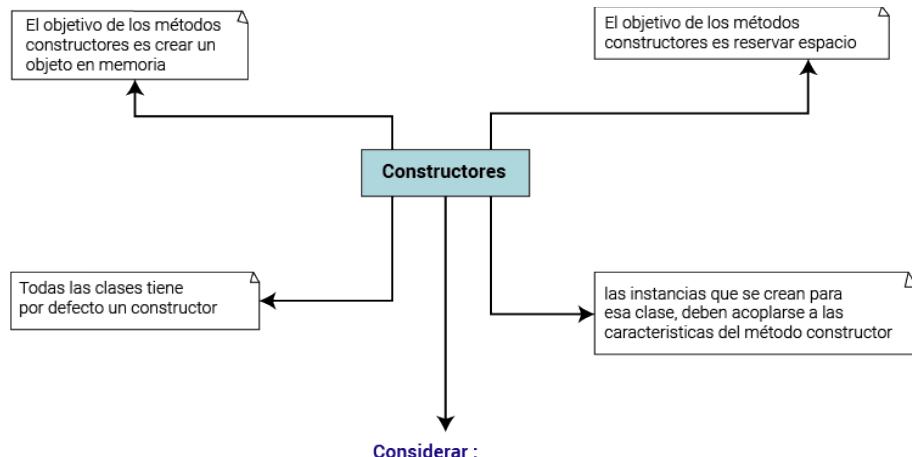


Figura 19. Nociones importantes sobre constructores.

Elaborado por: Elizalde, R. (2019)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En la guía didáctica se hace mención a dos casos para el manejo de constructores; se solicita leer cada una de las descripciones, además de analizar conjuntamente los ejemplos planteados.

- Para el caso 1: inicializar con valores específicos los datos de un objeto, se puede acceder al ejemplo desarrollado en miniespecificación y lenguaje de programación, a través de los siguientes enlaces.

Miniespecificación

[Desarrollo del caso planteado en miniespecificación](#)

Lenguaje de programación

[Desarrollo del caso planteado en lenguaje de programación.](#)

- Para el caso 2: inicializar el objeto mediante un constructor con valores enviados como parámetros, se puede acceder al ejemplo desarrollado en miniespecificación y lenguaje de programación, a través de los siguientes enlaces.

Miniespecificación

[Desarrollo del caso planteado sobre constructores en miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado sobre constructores en lenguaje de programación.](#)

Recursos de aprendizaje

- Analice los contenidos de la unidad 2. Creación de miniespecificaciones orientados a objetos y uso de constructores.

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 2 de la guía, brindan a los estudiantes ideas concretas sobre las temáticas relacionadas con la creación de miniespecificaciones y manejo de constructores en el paradigma de orientación a objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 9 "Programación Orientada a Objetos" aplicando la estructura de secuenciación; páginas 238-261.

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos a través de miniespecificaciones o pseudocódigo siguiendo el proceso de definición, análisis y diseño de un problema; además con base en los conceptos estudiados en la unidad 1, se hace uso de diagramas de clases para representar las soluciones. Adicionalmente, en el recurso planteado se presenta información sobre constructores en cuanto a su creación en una miniespecificación o diagrama de clases; y la implementación a través de ejercicios resueltos. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Revise el recurso colocado en el entorno virtual de aprendizaje sobre Creación de miniespecificaciones orientados a objetos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

LECTURA: Cachero, C., y Ponce de León, P. J. (2011). *Fundamentos de la programación orientada a objetos (2010-2011)*.

Programación orientada a objetos. Recuperado de <https://rua.ua.es/dspace/handle/10045/15994>

En el recurso se puede reforzar los conceptos referentes a la Programación Orientada a Objeto. Se detalla información relacionada a clases y objetos; atributos y operaciones de clases; y uso de constructores.

Resultados de aprendizaje 1 y 2

- Diseña, implementa, prueba y depura programas sencillos en un lenguaje de programación orientado a objetos.
- Describe cómo el mecanismo de la clase soporta la encapsulación y la ocultación de la información.

Contenidos, recursos y actividades de aprendizaje

Por lo indicado, es necesario entender y aprender los procedimientos para la creación de mini especificaciones y diagramas de clases en el ámbito de la Programación Orientada a Objetos y su relación con conceptos estudiados en ciclos académicos anteriores como las estructuras de decisión, que permitan generar soluciones robustas a problemáticas planteadas de la vida cotidiana.



Semana 4

2.3. Uso de estructuras de selección simple y compuesta en POO

¿Qué lecturas debo realizar para el estudio de la presente", sección?

Le solicitamos revisar:

- *En el texto básico el capítulo "Programación orientada a objetos aplicando las estructuras de selección", secciones: Diseño de algoritmos OO usando la selección doble (if-then-else) y Diseño de algoritmos OO usando la selección simple (if-then). Además, usted debe recordar los conceptos estudiados en el capítulo "La selección", secciones La selección doble (if-then-else), La selección simple (if-then) y La selección múltiple (switch).*
- *En la guía didáctica la unidad 2: Estructura y Creación de Algoritmos Orientados a Objeto , sección Uso de estructuras de selección simple y compuesta en POO.*

En la semana de trabajo se busca implementar los conceptos de estructuras de selección simple y doble en la generación de algoritmos orientados a objetos; se realizan ejemplos de diagramas de clases y su posterior implementación en miniespecificación, haciendo uso de las estructuras indicadas. En la guía didáctica se describe la implementación de las estructuras de selección simple y doble en el ambiente del paradigma de programación orientada a objetos. En la tabla número 4 dentro de la sección de la guía didáctica que estamos estudiando, se detallan las miniespecificaciones para las estructuras mencionadas.

A través de la figura 20, relacionamos la solución en miniespecificación y lenguaje de programación; en la misma se ha usado un condicional doble (ejemplo CALCULAR PENSIÓN DE UN ESTUDIANTE descrito en la guía didáctica). Se reitera el

pedido de revisar los ejemplos de la guía didáctica, en los ejemplos desarrollados se ha tomado la precaución de explicar las líneas de código más importantes, para una mejor compresión.

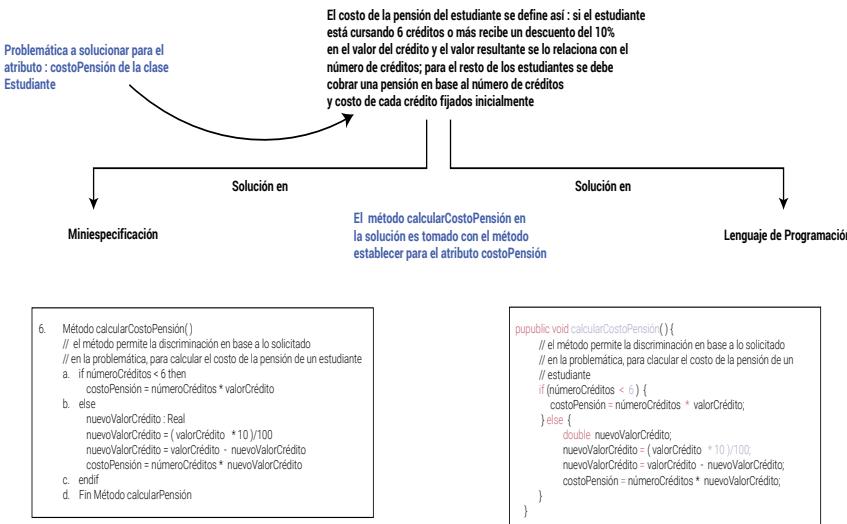


Figura 20. Uso de estructura de selección doble en programación orientada a objetos.

Elaborado por: Elizalde, R. (2019)

Es momento de explicar lo desarrollado en el método calcularCostoPension, a través de la figura 21.

Problemática a solucionar para el atributo: costoPensión de la clase Estudiante

El costo de la pensión del estudiante se define así: si el estudiante está cursando 6 créditos o más recibe un descuento del 10% en el valor del crédito y el valor resultante se lo relaciona con el número de créditos : para el resto de los estudiantes se debe cobrar una pensión en base al número de créditos y costo de cada crédito fijados inicialmente

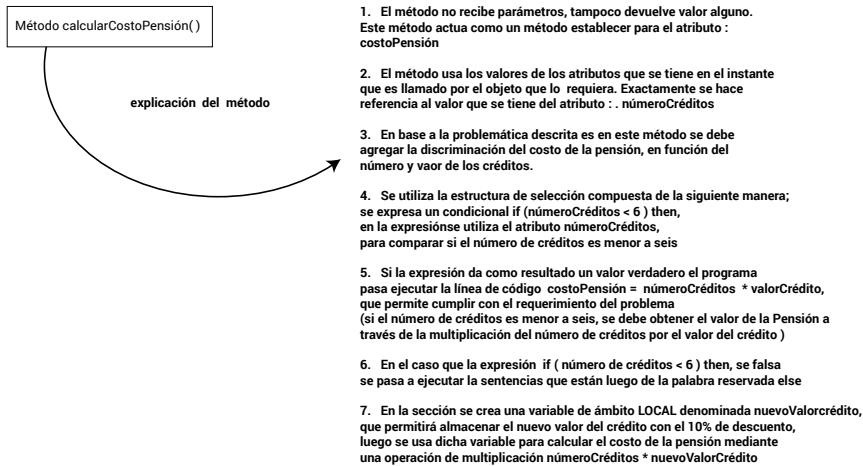


Figura 21. Explicación de la estructura de selección doble en el método.

Elaborado por: Elizalde, R. (2019)

En los siguientes enlaces se puede descargar (la solución en diagrama de clases, miniespecificación y lenguaje de programación) el ejemplo completo para la problemática: CALCULAR PENSIÓN DE UN ESTUDIANTE.

Diagrama

Desarrollo del caso planteado mediante diagrama

Miniespecificación

Desarrollo del caso planteado mediante miniespecificación

Lenguaje de Programación

Desarrollo del caso planteado mediante lenguaje de programación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Estimado estudiante, hemos llegado al fin de nuestra segunda unidad. Con la finalidad de retroalimentar sus conocimientos vamos a desarrollar la autoevaluación propuesta la unidad.

Recurso de aprendizaje

1. Analice los conceptos y ejercicios de la unidad 2. Uso de estructuras de selección simple y compuesta en POO.

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 2 de la guía, brindan a los estudiantes ideas concretas sobre las temáticas relacionadas con la generación de miniespecificaciones a través del uso de estructuras de selección simple y compuesta aplicando los conceptos de Programación Orientada a Objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 10 Programación Orientada a Objetos aplicando las estructuras de selección; páginas 264-282.

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos usando la selección doble (if-then-else), selección simple(if-then) a través de miniespecificaciones o pseudocódigo, además con base en los conceptos estudiados en la unidad 1, se hace uso de diagramas de clases para representar las soluciones. En la lectura se plantean ejercicios resueltos, que

tiene como finalidad indicar al estudiante en la práctica los conceptos estudiados en las unidades 1 y 2. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre Uso de estructuras de selección simple y compuesta en POO

LECTURA: Gómez R, J. (2015). *Control flow statements*. Recuperado de <http://ocw.uc3m.es/ingenieria-informatica/programming/class-material/control-flow-statements/>

El recurso pretende guiar al estudiante en el uso de estructuras de selecciones tanto simples como compuestas. Se hace uso de los conceptos estudiados, pero se agregar una explicación de implementación a través de un lenguaje de programación.



Actividades de aprendizaje recomendadas

Actividad 1

- **Actividad de aprendizaje**

La autoevaluación 2 presenta preguntas relacionadas con la estructura y creación de algoritmos orientados a objetos y el uso de estructuras de selección en POO. Se solicita revisar la unidad 2 de la guía de estudio y los capítulos “Programación Orientada a Objetos aplicando la estructura de secuenciación” y “Programación Orientada a Objetos aplicando las estructuras de selección” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, usted puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 2.



Autoevaluación 2

A continuación, se presentan preguntas relacionadas con la estructura y creación de algoritmos orientados a objetos y el uso de estructuras de selección en POO. Se solicita revisar la unidad 2 de la presente guía y los capítulos "Programación orientada a objetos aplicando la estructura de secuenciación" y "Programación orientada a objetos aplicando las estructuras de selección" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. ¿Cuál es la forma o sentencia correcta para empezar a desarrollar una miniespecificación?
 - a. Algoritmo *NOMBRE_DEL_ALGORITMO*
 - b. *NOMBRE_DEL_ALGORITMO*
 - c. *NOMBRE_DEL_ALGORITMO* Algoritmo

2. ¿Cuál es la función de los métodos "establecer"?
 - a. Asignar un valor a un dato de un objeto.
 - b. Recuperar el valor de un dato de un objeto
 - c. Imprimir el valor de un dato de un objeto.

3. ¿Cuál es la función de los métodos "obtener"?
- Recuperar el valor de un dato de un objeto.
 - Asignar un valor a un dato de un objeto
 - Imprimir el valor de un dato de un objeto.
4. Indique la forma correcta para declarar e inicializar un objeto.
- Clase1 c = Clase1()
 - Clase1 c = new Clase1()
 - c = new Clase1()
5. Con base en la siguiente clase:
- Declarar
Sueldo: Real
 - Método establecerSueldo(c: Real) sueldo = c
sueldo = c
 - Método obtenerSueldo()
return sueldo
Fin Método obtenerSueldo
Fin Clase Trabajador

Determine cuál de las siguientes sentencias son válidas y no generan error.

- Trabajor t = new Trabajador()
t.establecerSueldo(10.2)
- Trabajor t = Trabajador()
t.establecerSueldo(10.2)
- Trabajor t = new Trabajador()
t.obtenerSueldo(10.2)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

6. Con base en el concepto de constructores; indique la idea correcta del siguiente listado de expresiones.
- Las clases solo pueden tener 2 constructores.
 - Todas las clases tiene un constructor por defecto.
 - Los constructores tienen diferente nombre de la clase.
7. Dadas las siguientes clases.
- Declarar
placa: Cadena
 - Método Vehiculo()
placa = "LBBC-0183"
 - Método establecerPlaca(c: Cadena)
placa = c
Fin Método obtenerPlaca
Fin Clase Vehiculo
Clase Ejecuta
 - Método principal()
a. Vehiculo v = new Vehiculo()
b. Imprimir v.obtenerPlaca()
c. v.establecerPlaca("PBCC-0192")
d. Imprimir v.obtenerPlaca()
e. Fin Método principal

Determinar la salida que se obtendrá en pantalla.

- LBBC-0183
PBCC-0192
- PBCC-0192
LBBC-0183

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8. Con base en la siguiente clase:

Clase Vehiculo

1. Declarar
placa: Cadena
2. Método Vehiculo()
placa = "LBBC-0183"
3. Método establecerPlaca(pl: Cadena)
placa = pl
4. Método obtenerPlaca()
return placa
Fin Método obtenerPlaca
Fin Clase Vehiculo

Determine cuál de las siguientes sentencias generaría un error.

Opcion A

- Clase Ejecuta
1. Método principal()
 - a. Vehiculo v = new Vehiculo()
 - b. Fin Método principal
Fin Clase Ejecuta

Opcion B

- Clase Ejecuta
1. Método principal()
 - a. Vehiculo v = new Vehiculo("BDAS-001")
 - b. Fin Método principal
Fin Clase Ejecuta

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Opcion C

- Clase Ejecuta
1. Método principal()
 - a. Vehiculo v = new Vehiculo("ADEA-1234", "TGBA-4321")
 - b. Fin Método principal
 - Fin Clase Ejecuta
- a. Opción A.
 - b. Opción B.
 - c. Opción C.
9. Con base en las siguientes clases, indique la descripción que se ajuste a lo expresado en la miniespecificación.
- a. Si el usuario ingresa el valor de 1 en la variable opción, el programa declara e inicializa la clase **Vehículo** con el constructor que recibe un parámetro.
 - b. Si el usuario ingresa el valor de 1 en la variable opción, el programa declara e inicializa la clase **Vehículo** con el constructor por defecto.
 - c. Si el usuario ingresa el valor de 2 en la variable opción, el programa no puede declarar e inicializar la clase **Vehículo** de ninguna forma.
10. ¿Cuál es la forma correcta para declarar un constructor en miniespecificación?

Opción A

- Clase Edifico
1. Declarar
 2. Método Edificio()
 - Fin Método Edificio
 - Fin Clase Edificio

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Opción B

- Clase Edificio
- 1. Declarar
- 2. Método Contructor_Edificio()
- 3. Fin Método Contructor_Edificio
- Fin Clase Edificio

Opción C

- Clase Edificio
- 1. Declarar
- 2. Método Contructor()
- 3. Fin Método Contructor
- Fin Clase Edificio
- a. Opción A.
- b. Opción B.
- c. Opción C.

[Ir al solucionario](#)

Resultados de aprendizaje 1 y 3

- Diseña, implementa, prueba y depura programas sencillos en un lenguaje de programación orientado a objetos.
- Describe la relación entre la estructura estática de la clase y la estructura dinámica de las instancias de la clase.

Contenidos, recursos y actividades de aprendizaje

Para lo expuesto es necesario entender y aprender los procedimientos para la creación de miniespecificaciones y diagramas de clases en el ámbito de la Programación Orientada a Objetos y su relación con conceptos estudiados en ciclos académicos anteriores como las estructuras de repetición (do-while, while y for), que permiten generar soluciones robustas a problemáticas planteadas de la vida cotidiana.



Semana 5



Unidad 3. Estructuras de repetición en Programación Orientada a Objetos

3.1. Uso de estructura de repetición do-while en POO

Sigamos en el desarrollo de nuestra labor académica, para ello solicito revisar el contenido de:

- Texto básico en el capítulo Programación orientada a objetos aplicando las estructuras de repetición, sección Diseño de algoritmos OO usando la repetición do...while. Además, usted debe recordar los conceptos estudiados en el capítulo La repetición, sección La repetición do...while.
- Guía didáctica unidad 3: Estructuras de Repetición en Programación Orientada a Objetos, sección Uso de Estructura de repetición do-while en POO.

En la guía didáctica se explica el uso del ciclo repetitivo do-while a través de la resolución de la problemática planteada: se desea generar una solución informática que permita determinar el valor a pagar de la pensión mensual de un estudiante de un instituto educativo, en función de los créditos que está cursando en un ciclo en particular. El costo de la pensión del estudiante se define así: si el estudiante está cursando 6 créditos o más recibe un descuento del 10% en el valor del crédito y el valor resultante se lo relaciona con el número de créditos; para el resto de los estudiantes se debe cobrar una pensión en base al número de créditos y costo de cada

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

crédito fijados inicialmente. Hacer uso del ciclo repetitivo do-while para permitir el ingreso de varios estudiantes, el usuario decidirá el momento en que ya no desee ingresar más valores al proceso.

Estimado estudiante, usted puede acceder a la solución del problema a través de los siguientes enlaces:

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

[Desarrollo del caso planteado mediante miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado mediante lenguaje de programación.](#)

Lo anterior indica que se usará el ciclo repetitivo do-while en la clase que contenga el método principal (clase EjecutaEstudiante), la mencionada estructura permite determinar cuándo dejar de ingresar valores para ser asignados a los atributos de los objetos creados en cada iteración (objetos de tipo Estudiante). Vamos a explicar la clase EjecutaEstudiante, a través de la figura 22.

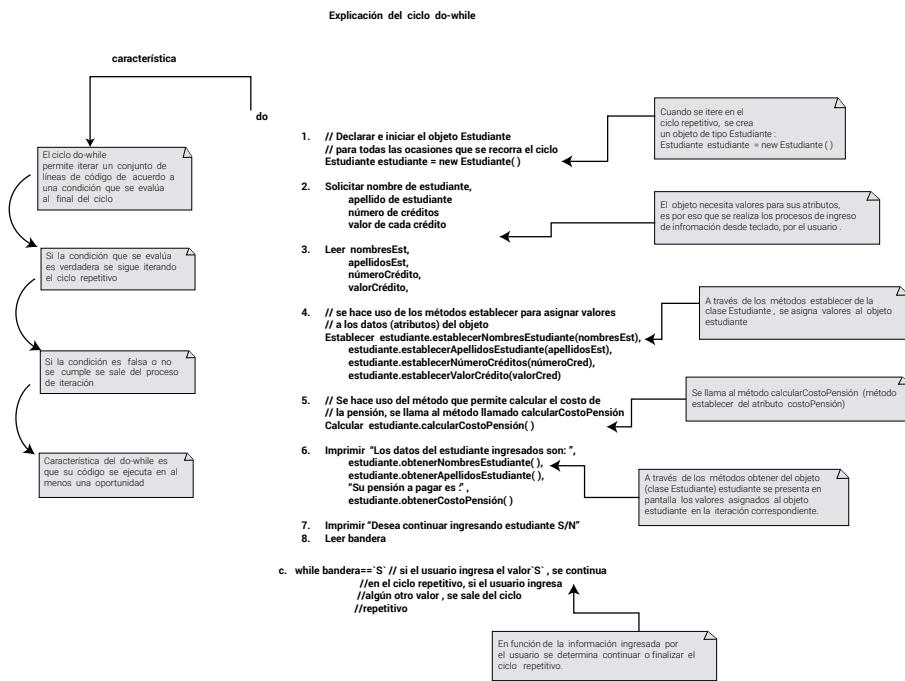


Figura 22. Explicación del ciclo repetitivo do-while.

Elaborado por: Elizalde, R. (2019)

3.2. Uso de Estructura de repetición while en POO

Para continuar con nuestra labor académica, se pide revisar los siguientes recursos:

- Texto básico en el capítulo Programación orientada a objetos aplicando las estructuras de repetición, sección Diseño de algoritmos OO usando la repetición while. Además, usted debe recordar los conceptos estudiados en el capítulo La repetición, sección La repetición while.

- *Guía didáctica unidad 3 “Estructuras de repetición en Programación Orientada a Objetos, sección Uso de estructura de repetición while en POO.*

De igual forma que en la estructura do-while, a través de una problemática se explica el funcionamiento de ciclo while aplicando conceptos de orientación a objetos. La problemática descrita en la guía didáctica está redactada en los siguientes términos: Se desea generar una solución informática que permita determinar el sueldo a pagar mensual de un docente de un instituto educativo, en función de las horas trabajadas en el mes y el costo por hora. Para obtener el valor a pagar se define lo siguiente: si el docente ha trabajado 40 horas o más, se le debe agregar a su sueldo una bonificación de \$150; para el resto de docente no se debe agregar ningún valor adicional

Estimado estudiante, tiene la posibilidad de descargar la solución a través de los siguientes enlaces.

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

[Desarrollo del caso planteado mediante miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado mediante lenguaje de programación.](#)

Se realiza una explicación del ciclo repetitivo while, usado en la clase EjecutaDocente; observar lo descrito en la figura 23.

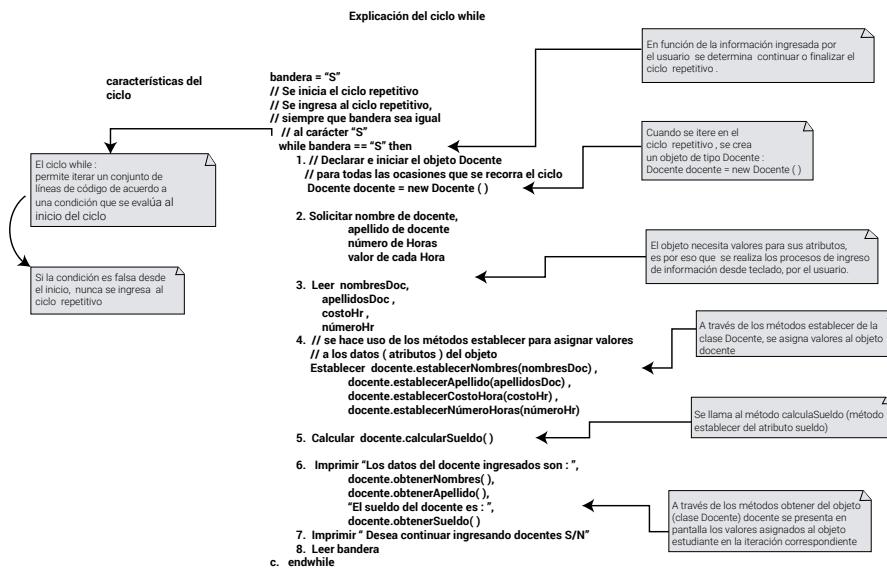


Figura 23. Explicación del ciclo repetitivo while.

Elaborado por: Elizalde, R. (2019)

Recurso de aprendizaje

- Estudie los contenidos y ejemplos de la unidad 3. Estructuras de repetición en Programación Orientada a Objetos

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 3 de la guía, brindan a los estudiantes ideas concretas sobre estructuras de repetición do-while y while en miniespecificaciones y diagramas de clases aplicando Programación Orientada a Objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- Revise los conceptos de la unidad 11. Programación Orientada a Objetos aplicando las estructuras de repetición; páginas 284 - 323

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos usando la repetición do-while y repetición while a través de miniespecificaciones o pseudocódigo; además, se hace énfasis en la importancia de manejar conceptos previamente estudiados como contadores y acumuladores. En la lectura se plantean ejercicios resueltos, que tiene como finalidad indicar al estudiante en la práctica los conceptos estudiados en las unidades 1, 2 y 3. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Visualice el recurso colocado en el entorno virtual de aprendizaje sobre Estructuras de Repetición en Programación Orientada a Objetos (do-while y while).

LECTURA: Gómez R, J. (2015). *Control flow statements*. Recuperado de <http://ocw.uc3m.es/ingenieria-informatica/programming/class-material/control-flow-statements/>

El recurso pretende guiar al estudiante en el uso de estructuras de repetitivas como el do-while y el while. Se hace uso de los conceptos sobre miniespecificaciones estudiados, pero se agregar una explicación de implementación a través de un lenguaje de programación.



Semana 6

3.3. Uso de Estructura de repetición for en POO

Para el estudio del presente apartado, diríjase a revisar el contenido de:

- Texto básico en el capítulo “Programación orientada a objetos aplicando las estructuras de repetición”, sección Diseño de algoritmos OO usando la repetición for. Además, usted debe recordar los conceptos estudiados en el capítulo “La repetición”, sección La repetición for.
- Guía didáctica unidad 3: Estructuras de Repetición en Programación Orientada a Objetos, sección Uso de Estructura de repetición for en POO.

En el apartado relacionado con el Uso de estructura de repetición for en Programación a Objetos en la guía didáctica, se plantea un ejemplo (**CALCULAR EL PROMEDIO DE CALIFICACIONES DE UN CONJUNTO DE ALUMNOS**), que necesita de un proceso repetitivo para su solución óptima.

El desarrollo de la solución para el problema lo puede descargar para su revisión y ejecución en su computador personal. Los enlaces son los siguientes.

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

Desarrollo del caso planteado mediante miniespecificación.

Lenguaje de programación

Desarrollo del caso planteado mediante lenguaje de programación.

El ciclo repetitivo for se requiere incluir en la clase EjecutaAlumno; en la figura 24 explicamos el proceso realizado.

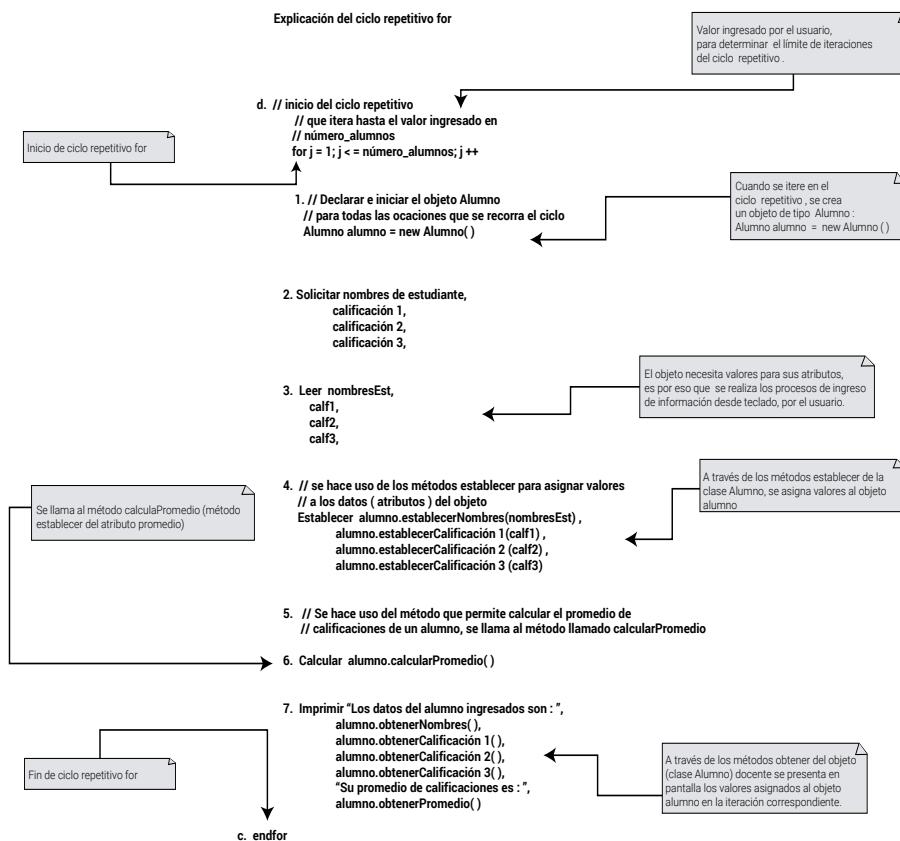


Figura 24. Explicación del ciclo repetitivo for en Programación Orientada a Objetos.

Elaborado por: Elizalde, R. (2019)

Recursos de aprendizaje

- Estudie los contenidos y ejemplos de la unidad 3. Estructuras de Repetición en Programación Orientada a Objetos

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 3 de la guía, brindan a los estudiantes ideas concretas sobre la estructura de repetición for en miniespecificaciones y diagramas de clases aplicando Programación Orientada a Objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos de la unidad 11. Programación Orientada a Objetos aplicando las estructuras de repetición; páginas 284 - 323

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos usando la for, a través de miniespecificaciones o pseudocódigo. En la lectura se plantean ejercicios resueltos, que tiene como finalidad indicar al estudiante en la práctica los conceptos estudiados en las unidades 1, 2 y 3. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Visualice el recurso colocado en el entorno virtual de aprendizaje sobre Estructuras de Repetición en Programación Orientada a Objetos (for).

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

LECTURA: Gómez R. J. (2015). *Control flow statements*. Recuperado de <http://ocw.uc3m.es/ingenieria-informatica/programming/class-material/control-flow-statements/>

El recurso pretende guiar al estudiante en el uso de estructuras de repetitivas como el for. Se hace uso de los conceptos sobre miniespecificaciones estudiados, pero se agregar una explicación de implementación a través de un lenguaje de programación.



Actividades de aprendizaje recomendadas

Actividad 1

- **Actividad de aprendizaje**

La autoevaluación 3 presenta preguntas relacionadas con el manejo estructuras de repetición en Programación Orientada a Objetos. Se solicita revisar la unidad 3 de la presente guía y el capítulo "Programación Orientada a Objetos aplicando las estructuras de repetición" del texto básico para contestar las interrogantes. La autoevaluación permitirá a usted reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en cuanto a la autoevaluación 3.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Autoevaluación 3

A continuación, se presentan preguntas relacionadas con manejo estructuras de repetición en programación orientada a objetos. Se solicita revisar la unidad 3 de la presente guía y capítulo "Programación orientada a objetos aplicando las estructuras de repetición" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. Se necesita implementar una clase llamada **Ejecutar**, que permita instanciar 5 objetos de tipo **Docente**, a cada objeto se le debe asignar un valor para su atributo o dato edad, haciendo uso del método **establecerEdad**.

Dada una clase:

Clase Docente

1. Declarar
edad: Entero
2. Método establecerEdad(c: Entero)
edad = c
Fin Método establecerEdad
3. Método obtenerEdad()
return edad
Fin Método obtenerEdad
Fin Clase Docente

Identifique la miniespecificación correcta:

Opción A

Clase Ejecutar
1. Método principal()
a. Declarar Variables
e: Entero
contador: Entero
contador = 1
b. while contador < 5 then
Docente d = new Docente()
Solicitar edad
Leer e
d.establecerEdad(e)
contador = contador + 1
endwhile
Fin Método principal
Fin Clase Ejecutar
Fin

Opción B)

Clase Ejecutar
1. Método principal()
a. Declarar Variables
e: Entero
contador: Entero
contador = 1
b. while contador < 5 then
Docente d = new Docente()
Solicitar edad
Leer e
d.establecerEdad(e)

```
contador = contador + 1
endwhile
Fin Método principal
Fin Clase Ejecutar
Fin
```

Opción C

- Clase Ejecutar
1. Método principal()
 - a. Declarar Variables
e: Entero
contador: Entero
contador = 1
 - b. while contador <= 5 then
Docente d = new Docente()
Solicitar edad
Leer e
d.establecerEdad(e)
contador = contador + 1
endwhile
Fin Método principal
Fin Clase Ejecutar
Fin
 - a. Opción A
b. Opción B
c. Opción C
2. Dadas las siguientes clases:
- Clase Docente
1. Declarar
edad: Entero
 2. Método establecerEdad(c: Entero)

```
edad = c
Fin Método establecerEdad
3. Método obtenerEdad()
    return edad
    Fin Método obtenerEdad
    Fin Clase Docente

    Clase Ejecutar
1. Método principal()
a. Declarar Variables
    e: Entero
    contador: Entero
    contador = 1
b. do
    Docente d = new Docente()
    Solicitar edad
    Leer e
    d.establecerEdad(e)
    contador = contador + 1
    while contador < 1
        Fin Método principal
        Fin Clase Ejecutar
        Fin
```

Indique la descripción que se ajuste a lo expresado en la miniespecificación.

- a. El programa permite crear dos objetos de tipo Docente.
- b. El programa permite crear un objeto de tipo Docente.
- c. El programa no permite crear ningún objeto de tipo Docente.

3. Dada la siguiente clase:

```
Clase Trabajador
1. Declarar
    sueldo: Real

2. Método establecerSueldo(c: Real)
    sueldo = c
    Fin Método establecerEdad

3. Método obtenerSueldo()
    return sueldo
    Fin Método obtenerSueldo
    Fin Clase Trabajador
```

Se requiere identificar la miniespecificación correcta, que permita crear mediante un ciclo repetitivo uno o varios objetos de tipo Trabajador.

Opción A

```
Clase Ejecutar
1. Método principal()
a. Declarar Variables
    e: Real
b. for i=5; i<5;i++
    Trabajador d = new Trabajador()
    Solicitar sueldo
    Leer e
    d.establecerSueldo(e)
    contador = contador + 1
    endfor
    Fin Método principal
    Fin Clase Ejecutar
    Fin
```

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Opción B

```
Clase Ejecutar
1. Método principal()
a. Declarar Variables
e: Real
contador: Entero
contador = 5
b. while contador <5
Trabajdor d = new Trabajador()
Solicitar sueldo
Leer e
d.establecerSueldo(e)
contador = contador + 1
endwhile
Fin Método principal
Fin Clase Ejecutar
Fin
```

Opción C

```
Clase Ejecutar
1. Método principal()
a. Declarar Variables
e: Real
contador: Entero
contador = 1
b. do
Trabajdor d = new Trabajador()
Solicitar sueldo
Leer e
d.establecerSueldo(e)
contador = contador + 1
while contador < 5
Fin Método principal
```

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Fin Clase Ejecutar

Fin

- a. Opción A
- b. Opción B
- c. Opción C

4. Dada la siguiente clase:

Clase Trabajador

- 1. Declarar
sueldo: Real
- 2. Método establecerSueldo(c: Real)
sueldo = c
Fin Método establecerEdad
- 3. Método obtenerSueldo()
return sueldo
Fin Método obtenerSueldo
Fin Clase Trabajador

Se requiere identificar la miniespecificación correcta, que permita crear mediante un ciclo repetitivo uno o varios objetos de tipo Trabajador y permita asignar valores al atributo sueldo.

Opción A

Clase Ejecutar

- 1. Método principal()
- a. Declarar Variables
e: Real
- b. for i=0; i<5;i++
Trabajador d = new Trabajador()
Solicitar sueldo
Leer e
d.obtenerSueldo(e)
endfor

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Fin Método principal

Fin Clase Ejecutar

Fin

Opción B

Clase Ejecutar

1. Método principal()

a. Declarar Variables

e: Real

b. for i=0; i<5;i++

Trabajador d = new Trabajador()

Solicitar sueldo

Leer e

d.establecerSueldo(e)

endfor

Fin Método principal

Fin Clase Ejecutar

Fin

Opción C

Clase Ejecutar

1. Método principal()

a. Declarar Variables

e: Real

b. for i=0; i<5;i++

Trabajador d = new Trabajador()

endfor

Fin Método principal

Fin Clase Ejecutar

Fin

a. Opción A.

b. Opción B.

c. Opción C.

5. Dadas las siguientes clases:

```
Clase Trabajador
1. Declarar
    sueldo: Real

2. Método establecerSueldo(c: Real)
    sueldo = c
    Fin Método establecerEdad

3. Método obtenerSueldo()
    return sueldo
    Fin Método obtenerSueldo
    Fin Clase Trabajador
    Clase Ejecutar
1. Método principal()
a. Declarar Variables
    e: Real
    opcion: Entero
    contador: Entero
b. Solicitar opcion
    Leer opcion
    contador = 1
    if opcion == 1 then
        while contador <=3
            Trabajdor d = new Trabajador()
            Solicitar sueldo
            Leer e
            d.obtenerSueldo(e)
            contador = contador + 1
        endwhile
    else
        while contador <3
            Trabajdor d = new Trabajador()
            Solicitar sueldo
```

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

```
Leer e  
d.obtenerSueldo(e)  
contador = contador + 1  
endwhile  
endif  
Fin Método principal  
Fin Clase Ejecutar  
Fin
```

Indique la descripción que se ajuste a lo expresado en la miniespecificación.

- a. Si el usuario ingresa la opción 1, a través de un ciclo repetitivo creará tres objetos de tipo Trabajador; caso contrario, ingresar a través de un ciclo repetitivo cuatro objetos tipo Trabajador.
- b. Si el usuario ingresa la opción 1, a través de un ciclo repetitivo creará dos objetos de tipo Trabajador; caso contrario, ingresar a través de un ciclo repetitivo tres objetos tipo Trabajador.
- c. Si el usuario ingresa la opción 1, a través de un ciclo repetitivo creará tres objetos de tipo Trabajador; caso contrario, ingresar a través de un ciclo repetitivo dos objetos tipo Trabajador.

[Ir al solucionario](#)

Muy bien, hemos avanzado con gran parte de nuestro estudio, ahora es tiempo de preparar la evaluación presencial del primer bimestre.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Actividades finales del bimestre



Semana 7



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

La autoevaluación 1 se plantea a través de preguntas relacionadas con Programación Orientada a Objetos usando diagramas de clase. Se solicita revisar la unidad 1 de la presente guía y el capítulo “Programación Orientada a Objetos usando el diagrama de clases” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en cuanto a la autoevaluación 1.

Actividad 2

- **Actividad de aprendizaje**

La autoevaluación 2 presenta preguntas relacionadas con la estructura y creación de algoritmos orientados a objetos y el uso de estructuras de selección en POO. Se solicita revisar la unidad 2 de la guía de estudio y los capítulos "Programación Orientada a Objetos aplicando la estructura de secuenciación" y "Programación Orientada a Objetos aplicando las estructuras de selección" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la Autoevaluación 2.

Actividad 3

- **Actividad de aprendizaje**

Revisión de la autoevaluación 3, donde se presentan preguntas relacionadas con manejo estructuras de repetición en Programación Orientada a Objetos. Se solicita revisar la unidad 3 de la presente guía y el capítulo "Programación Orientada a Objetos aplicando las estructuras de repetición" del texto básico para contestar las interrogantes. La autoevaluación permitirá a usted reafirmar los conceptos estudiados.

■ Procedimiento

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 3.



Semana 8

Actividad 1

Las semana 8 se define como un espacio de tiempo que permite a los estudiantes reforzar temáticas no comprendidas en las primeras semanas de estudio, de cara a la evaluación presencial del primer bimestre; por ello, se propone el repaso de las unidades de estudio revisadas hasta el momento.

- Unidad 1. Introducción a Programación Orientada a Objetos.
- Unidad 2. Estructura y creación de algoritmos orientados a objetos.
- Unidad 3. Estructuras de repetición en Programación Orientada a Objetos.

La revisión consiste en realizar las siguientes acciones.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- Revisar los recursos de aprendizaje recomendados en cada semana.
- Realizar nuevamente las autoevaluaciones de cada unidad del primer bimestre.
- Revisar las preguntas de los cuestionarios de refuerzo.
- Plantearse ejercicios y resolverlos.



Segundo bimestre

Resultados de aprendizaje 1, 2 y 5

- Diseña, implementa, prueba y depura programas sencillos en un lenguaje de programación orientado a objetos.
- Describe cómo el mecanismo de la clase soporta la encapsulación y la ocultación de la información.
- Utiliza iteradores para acceder a los elementos de un contenedor.

Contenidos, recursos y actividades de aprendizaje

En las unidades del primer bimestre de la asignatura se establecieron bases en el aprendizaje del paradigma de programación de orientación a objetos. El inicio del segundo bimestre se plantea el estudio de estructuras de datos simples y de objetos; a través de la revisión de conceptos, ligados al desarrollo de ejercicios prácticos usando diagramas de clases y miniespecificaciones.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En la semana 9 de actividades del segundo bimestre se estudia la unidad 4 relacionada con Estructuras de Datos en Programación Orientada a Objetos; los subtemas son:

- Uso de Arreglos de Tipos de Datos Simples o Primitivos.
- Uso de Arreglos de Tipos de Datos de Objetos.

Las temáticas indicadas permiten a los estudiantes alcanzar nociones importantes en el manejo de estructuras de datos e información, a través de la explicación del uso de arreglos con tipos de datos simples y tipos de datos de objetos; se hace énfasis en la creación de diagramas de clase y miniespecificaciones.



Semana 9



Unidad 4. Estructuras de datos en Programación Orientada a Objetos

A continuación, analizaremos aspectos muy interesantes de la programación orientada a objetos, para ello se requiere revisar el contenido de:

- Texto básico en el capítulo "Programación orientada a objetos aplicando arreglos", secciones: Diseño de algoritmos OO

usando arreglos unidimensionales y Diseño de algoritmos OO usando arreglos bidimensionales.

- Guía didáctica unidad 4, "Estructuras de Datos en Programación Orientada a Objetos" secciones: Uso de Arreglos de Tipos de Datos Simples o Primitivos y Uso de Arreglos de Tipos de Datos de Objetos.

4.1. Uso de arreglos de tipos de datos simples o primitivos

En la guía didáctica se realiza la explicación de un ejemplo de una clase o entidad denominada Paralelo, que tiene atributos o datos de tipo arreglo. Los archivos solución para el ejemplo en mención están en los siguientes enlaces. Por favor, revisarlos y ejecutarlos en sus máquinas personales.

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

[Desarrollo del caso planteado mediante miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado mediante lenguaje de programación.](#)

A continuación, en la figura 25 se detallan algunas consideraciones importantes a tomar en cuenta para futuras problemáticas que sugieren en su análisis el uso de tipos de datos arreglos.

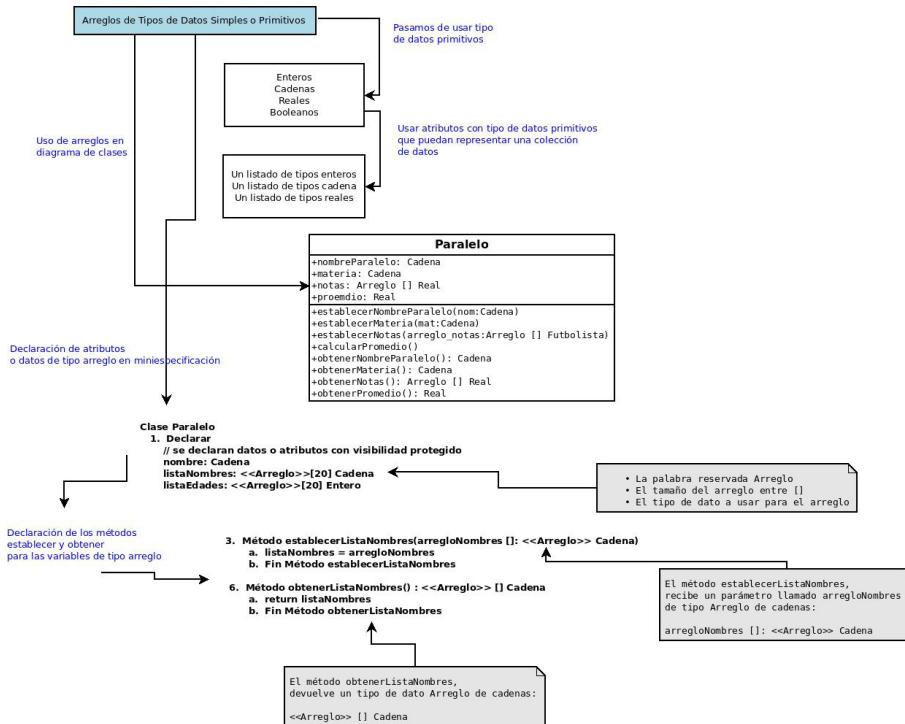


Figura 25. Consideraciones importantes para el uso de arreglos en Programación Orientada a Objetos.

Elaborado por: Elizalde, R. (2019)

4.2. Uso de arreglos de tipos de datos de objetos

Se recomienda revisar el ejemplo de la sección Uso de arreglos de tipo de datos de objetos de la guía didáctica; donde se describe la solución para crear atributos o datos de una clase que haga referencia a otras clases.

El ejemplo completo usted lo puede revisar y descargar en los siguientes enlaces.

Diagrama

Desarrollo del caso planteado mediante diagrama.

Miniespecificación

Desarrollo del caso planteado mediante miniespecificación.

Lenguaje de programación

Desarrollo del caso planteado mediante lenguaje de programación.

En la figura 26 explicamos algunas pautas para tomar en consideración cuando se tenga problemáticas similares.

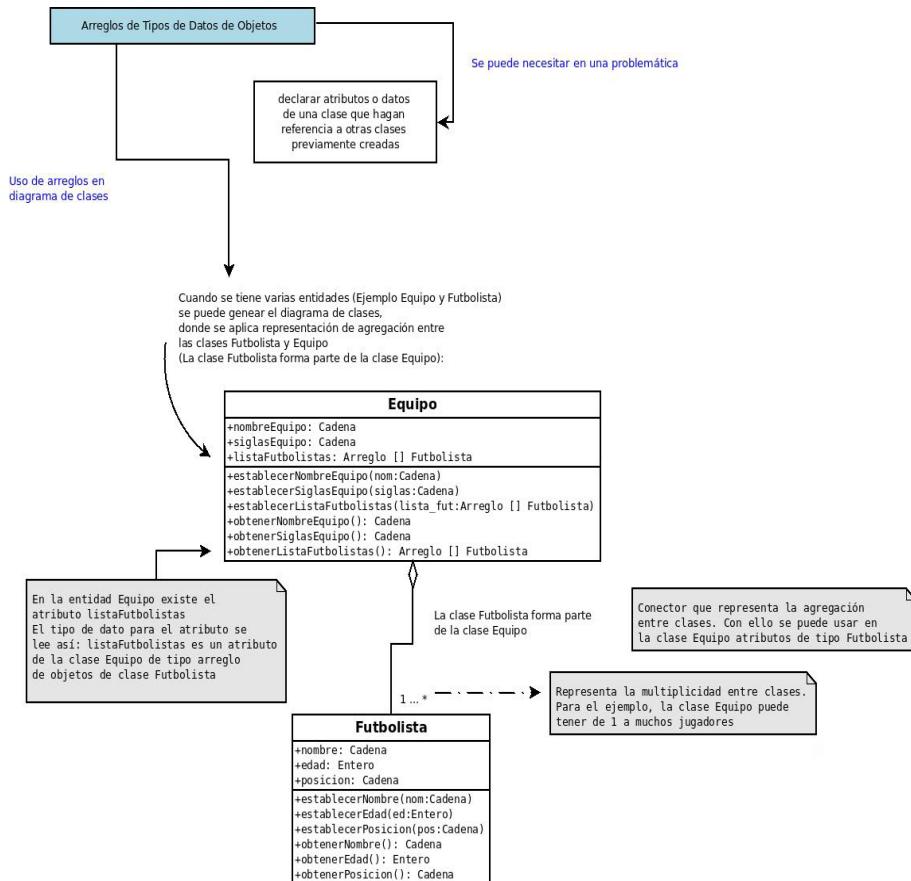


Figura 26. Arreglos de tipo de datos de objetos.

Elaborado por: Elizalde, R. (2019)

Recursos de aprendizaje

- Lea los contenidos y ejemplos de la unidad 4. Estructuras de datos en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). *Guía didáctica de Programación Orientada a Objetos*. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 4 de la guía, permiten a los estudiantes conocer sobre el manejo de estructuras de datos simple y de objetos a través de miniespecificaciones y diagramas de clases aplicando Programación Orientada a Objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejemplos de la unidad 12 Programación Orientada a Objetos aplicando arreglos; páginas 326-355.

LECTURA: López, L. (2013). *Metodología de la Programación Orientada a Objetos*. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos usando arreglos unidimensionales para estructuras de datos simples y de objetos, a través de la construcción de miniespecificaciones y diagramas de clases para representar las soluciones. La lectura explica ejercicios resueltos, que tiene como finalidad indicar al estudiante en la práctica los conceptos estudiados en la unidad. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Estructuras de Datos en Programación Orientada a Objetos - Uso de Arreglos de Tipos de Datos Simples o Primitivos

LECTURA: García, A. y Arranz, J. (2010). Algunas clases estándar de Java (II). Recuperado de http://ocw.upm.es/pluginfile.php/1037/mod_label/intro/11-algunasclasesestandardejavaii.pdf

El recurso ayudará al estudiante en el entendimiento y posterior exemplificación de estructuras de datos en problemas de programación. Además permite ser una guía referencial de implementación de estructuras en un lenguaje de programación.



Semana 10

4.3. Ejercicios propuestos usando arreglos

Revisar el desarrollo de los ejercicios planteados la guía didáctica en la unidad 4 "Estructuras de Datos en Programación Orientada a Objetos", sección Ejercicios propuestos usando arreglos. Luego de abordar los contenidos sobre el manejo de estructuras a través de arreglos y su implementación en diagramas y miniespecificación; es necesario crear y explicar ejemplos de las temáticas, la idea fundamental es explicar detenidamente el funcionamiento de arreglos con tipos de datos simples o primitivos, y datos de objetos; asociados a los conceptos de Programación Orientada a Objetos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Estimado estudiante: a partir de ahora revisaremos más detenidamente ejemplos de arreglos con tipos de datos simples o primitivos y datos de objetos.

4.3.1. Arreglos con tipos de datos simples o primitivos

Se solicita revisar, analizar y ejecutar en sus computadoras personales el ejemplo descrito en la guía didáctica denominado: **PROMEDIO DE NOTAS DE UN PARALELO**, donde se hace uso de un arreglo de tipo de datos simples o primitivos.

Los enlaces para descarga de los archivos de solución son los siguientes.

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

[Desarrollo del caso planteado mediante miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado mediante lenguaje de programación.](#)

4.3.2. Arreglos con tipos de datos de objetos

Bien, ahora se pide la revisión, análisis y ejecución en sus computadoras personales del ejemplo descrito en la guía didáctica denominado: **PROMEDIO DE EDADES DE LOS ESTUDIANTES DE UN PARALELO**, donde se hace uso de un arreglo de tipo de datos de objetos

Los enlaces para descarga de los archivos solución son los siguientes:

Diagrama

Desarrollo del caso planteado mediante diagrama.

Miniespecificación

Desarrollo del caso planteado mediante miniespecificación.

Lenguaje de programación

Desarrollo del caso planteado mediante lenguaje de programación.

Ahora, para finalizar el estudio de los apartados anteriores, se deja el planteamiento y desarrollo de la siguiente problemática.

Se requiere implementar un diagrama, miniespecificación e implementación en lenguaje de alto nivel (Java), para una clase llamada Libro. Dicha entidad tendrá dos atributos:

- El nombre del libro.
- El listado de autores del libro.

Implementación en diagrama de clases.

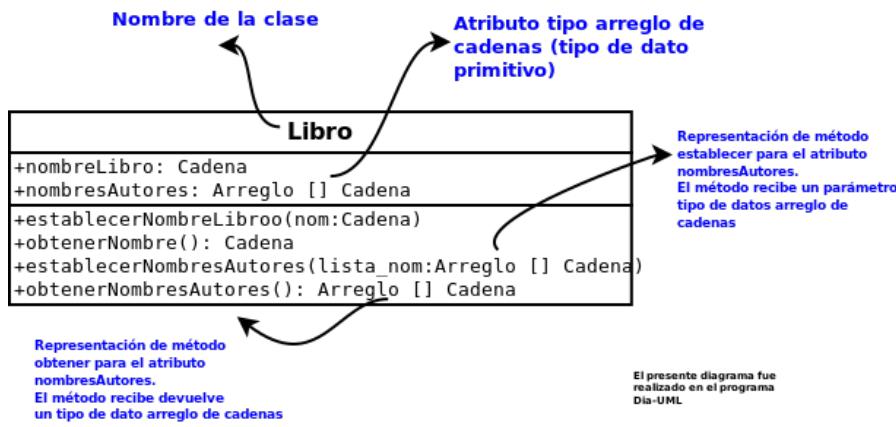


Figura 27. Arreglos de tipo de datos de objetos.

Elaborado por: Elizalde, R. (2019)

Implementación en miniespecificación

- Enlace para descarga y visualización del código en miniespecificación

Implementación en lenguaje de programación

- Enlace para descarga y visualización del código implementado en lenguaje de programación Java

Recursos de aprendizaje

- Lea los contenidos y ejemplos de la unidad 4. Estructuras de Datos en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En la segunda parte de los contenidos de la unidad 4 de la guía, se plantean ejercicios que permiten a los estudiantes entender el procedimiento de creación de soluciones haciendo uso de estructuras de datos simples y de objetos. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejemplos de la unidad 12. Programación Orientada a Objetos aplicando arreglos; páginas 326 - 355

LECTURA: López, L. (2013). Metodología de la Programación Orientada a Objetos. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de algoritmos de Programación Orientada a Objetos usando arreglos unidimensionales para estructuras de datos simples y de objetos, a través de la construcción de miniespecificaciones y diagramas de clases para representar las soluciones. La lectura explica ejercicios resueltos, que tiene como finalidad indicar al estudiante en la práctica los conceptos estudiados en la unidad. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Estructuras de Datos en Programación Orientada a Objetos - Uso de Arreglos de Tipos de Datos Simples o Primitivos

LECTURA: García, A. y Arranz, J. (2010). Algunas clases estándar de Java (II). Recuperado de http://ocw.upm.es/plugin_ile.php/1037/mod_label/intro/11-algunasclasesestandardejavaii.pdf

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

El recurso ayudará al estudiante en el entendimiento y posterior exemplificación de estructuras de datos en problemas de programación. Además permite ser una guía referencial de implementación de estructuras en un lenguaje de programación.



Actividades de aprendizaje recomendadas

Actividad 1

- **Actividad de aprendizaje**

La autoevaluación 4 se plantea a través de preguntas relacionadas al manejo de arreglos en Programación Orientada a Objetos.

Se solicita revisar la unidad 4 de la presente guía y el capítulo "Programación Orientada a Objetos aplicando arreglos" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 4.



Autoevaluación 4

A continuación, se presentan preguntas relacionadas al manejo de arreglos en programación orientada a objetos. Se solicita revisar la unidad 4 de la presente guía y el capítulo "Programación Orientada a Objetos aplicando arreglos" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. ¿Cuál es el diagrama de clases correcto para una clase que tenga entre sus datos o atributos un arreglo que permita almacenar los meses del año?

Opción A

Clase
+descripcion: Cadena
+meses: Arreglo [12] Cadena

Opción B

Clase
+descripcion: Cadena
+meses: Arreglo [12]

Opción C

Clase
+descripcion: Cadena
+mes1: Cadena
+mes2: Cadena
+...
+mes12: Cadena

- a. Opción A
- b. Opción B
- c. Opción C

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

2. ¿Cuál es la miniespecificación correcta para una clase que tenga entre sus datos o atributos un arreglo que permita almacenar los meses del año?

Opción A

```
Clase Clase
Declarar
// se declaran datos o atributos
descripción: Cadena
meses: <>Arreglo>>[12]
Fin Clase Paralelo
```

Opción B

```
Clase Clase
Declarar
// se declaran datos o atributos
descripción: Cadena
mes1: Cadena
mes2: Cadena
...
mes12: Cadena
Fin Clase Paralelo
```

Opción C

```
Clase Clase
Declarar
// se declaran datos o atributos
descripción: Cadena
meses: <>Arreglo>>[12] Cadena
Fin Clase Paralelo
```

- a. Opción A
- b. Opción B
- c. Opción C

3. En una clase se tiene un atributo de tipo arreglo de enteros, denominado promedios. Identifique cuál es la miniespecificación correcta, que permita generar los métodos establecer y obtener para el atributo mencionado en dicha clase.

Opción A

```
Clase Clase
1. Declarar
    // se declaran datos o atributos
    promedios: <<Arreglo>>[12] Entero
2. Método establecerPromedios(lista[]: <<Arreglo>> Entero)
    promedios = lista
    Fin establecerPromedios
3. Método obtenerPromedios()
    return promedios
    Fin obtenerPromedios
    Fin Clase Clase
```

Opción B

```
Clase Clase
1. Declarar
    // se declaran datos o atributos
    promedios: <<Arreglo>>[12] Entero
2. Método establecerPromedios(lista[]: <<Arreglo>> Entero)
    promedios = lista[0]
    Fin establecerPromedios
3. Método obtenerPromedios()
    return promedios[0]
    Fin obtenerPromedios
    Fin Clase Clase
```

Opción C

- Clase Clase
1. Declarar
// se declaran datos o atributos
promedios: <>Arreglo>>[12] Entero
 2. Método establecerPromedios(lista[]: <>Arreglo>> Cadena)
promedios = lista
Fin establecerPromedios
 3. Método obtenerPromedios()
return lista
Fin obtenerPromedios
Fin Clase Clase
- a. Opción A
b. Opción B
c. Opción C
4. Se desea encontrar la suma de los datos ingresados para el atributo valores. Se ha generado la siguiente miniespecificación, en la misma falta implementar el método calcularSuma. Se solicita identificar la implementación correcta del método calcularSuma.

- Clase Clase1
1. Declarar
// se declaran datos o atributos
valores: <>Arreglo>>[3] Entero
suma: Entero
 2. Método establecerValores(listaValores[]: <>Arreglo>>
Cadena)
valores = listaValores
Fin Método establecerValores

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

3. Método obtenerValores()
return valores
Fin Método obtenerValores
4. Método calcularSuma()
// Encontrar la implementación para este método
Fin Método calcularSuma
5. Método obtenerSuma()
return suma
Fin Método obtenerValores
Fin Clase Clase1
Clase Ejecutar
 1. Método principal()
 - a. Declarar Variables
lista_valores: <>Arreglo>>[3] Entero
 - b. for i = 0; i<3; i++
 1. Solicitar lista_valores[i]
 - c. Leer lista_valores[i]
 - d. endfor
 - e. Clase1 c = new Clase1()
 - f. Establecer:
clase.establecerValores(lista_valores)
 - g. Calcular: c.calcularSuma()
 - h. Imprimir "La suma de los valores es",
c. obtenerSuma(),
 2. Fin Método principal
 3. Fin Clase Ejecutar
 4. Fin

Implementaciones del método calcularSuma.

Opción A

```
4. Método calcularSuma()
a. suma = 0
for i = 0; i<3; i++
1. suma = suma + valores[i]
endfor
Fin Método calcularSuma
```

Opción B

```
4. Método calcularSuma()
a. suma = 0
for i = 0; i<3; i++
1. suma = suma + valores[0]
endfor
Fin Método calcularSuma
```

Opción C

- 4. Método calcularSuma()
 - a. suma = 0
 - for i = 0; i<3; i++
 - 1. suma = valores[0]
 - endfor
 - Fin Método calcularSuma
- a. Opción A
- b. Opción B
- c. Opción C

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. Dadas las siguientes clases, identifique la implementación correcta para la clase Ejecutar, que permita asignar valores correctamente al atributo valores de la Clase1.

Clase Docente

1. Declarar
edad: Entero
 2. Método establecerEdad(c: Entero)
edad = c
Fin Método establecerEdad
 3. Método obtenerEdad()
return edad
Fin Método obtenerEdad
- Clase Clase1
1. Declarar
// se declaran datos o atributos
valores: <<Arreglo>>[3] Docente
 2. Método establecerValores(listaValores[]:
<<Arreglo>> Docente)
valores = listaValores
Fin Método establecerValores
 - Método obtenerValores()
return valores
Fin Método obtenerValores
 - Fin Clase Clase1

Implementaciones de la clase Ejecutar.

Opción A

Clase Ejecutar

1. Método principal()
 - a. Declarar Variables
lista_valores: <<Arreglo>>[3] Docente
 - b. Docente d = new Docente()

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

```
d.establecerEdad(30)
Docente d2 = new Docente()
d2.establecerEdad(32) Docente d3 = new Docente()
d3.establecerEdad(31)
lista_valores[0] = d
lista_valores[1] = d2
lista_valores[2] = d3
Clase1 c1 = new Clase1()
c1.establecerValores(lista_valores)
Fin Método principal
Fin Clase Ejecutar
Fin
```

Opción B

```
Clase Ejecutar
1. Método principal()
a. Declarar Variables
lista_valores: <>Arreglo>>[3] Docente
b. Docente d = new Docente()
d.establecerEdad(30)
Docente d2 = new Docente()
d2.establecerEdad(32)
Docente d3 = new Docente()
d3.establecerEdad(31)
lista_valores[0] = d
lista_valores[1] = d2
lista_valores[2] = d3
Clase1 c1 = new Clase1()
c1.establecerValores(d1, d2, d3)
Fin Método principal
Fin Clase Ejecutar
Fin
```

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Opción C

Clase Ejecutar

```
1. Método principal()
   a. Declarar Variables
      lista_valores: <<Arreglo>>[3] Docente
      b. Docente d = new Docente()
      d.establecerEdad(30)
      Docente d2 = new Docente()
      d2.establecerEdad(32)
      Docente d3 = new Docente()
      d3.establecerEdad(31)
      lista_valores[0] = d
      lista_valores[1] = d2
      lista_valores[2] = d3
      Clase1 c1 = new Clase1()
      c1.establecerValores(lista_valores[2])
      Fin Método principal
      Fin Clase Ejecutar
      Fin
```

- a. Opción A
- b. Opción B
- c. Opción C

[Ir al solucionario](#)

Felicito su empeño y dedicación, ahora es momento de iniciar una nueva unidad de estudio.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultados de aprendizaje 1, 3 ,7 y 8

- Diseña, implementa, prueba y depura programas sencillos en un lenguaje de programación orientado a objetos.
- Describe la relación entre la estructura estática de la clase y la estructura dinámica de las instancias de la clase.
- Discute e identifica los conceptos de encapsulación, abstracción, herencia y polimorfismo.
- Diseña, implementa y prueba la implementación de una relación “is-a” entre objetos usando una jerarquía herencia de clases.

Contenidos, recursos y actividades de aprendizaje



Semana 11

Para que el estudiante pueda completar los resultados esperados, se requiere el estudio a nivel teórico y práctico, sobre las directrices necesarias para implementar la herencia en una problemática

específica; se dará importancia al desarrollo de diagramas de clase y miniespecificaciones; sentando las bases para que el estudiante en lo posterior pueda implementar soluciones en lenguajes de alto nivel.

Hemos llegado a un apartado tan interesante como importante, el estudio de herencia, superclase y subclase. Para ello se solicita revisar el contenido de:

- Texto básico en el capítulo “Programación orientada a objetos usando herencia”, secciones Herencia, Diseño del diagrama de clases con herencia y Diseño de algoritmos OO usando herencia.
- Guía didáctica unidad 5 Herencia en Programación Orientada a Objetos secciones Concepto de Herencia - Superclases y Subclases, Uso de Diagramas de Clase con Herencia.



Unidad 5. Herencia en Programación Orientada a Objetos

5.1. Concepto de herencia

Un concepto fundamental en el paradigma de Programación Orientada a Objetos es el de herencia, en la semana 3 de actividades académicas se estudia: la relación entre superclases-subclases y la implementación de dicha relación en diagramas de clases.

En relación con lo explicado en la guía didáctica se exponen las siguientes características del concepto de herencia (figura 28).

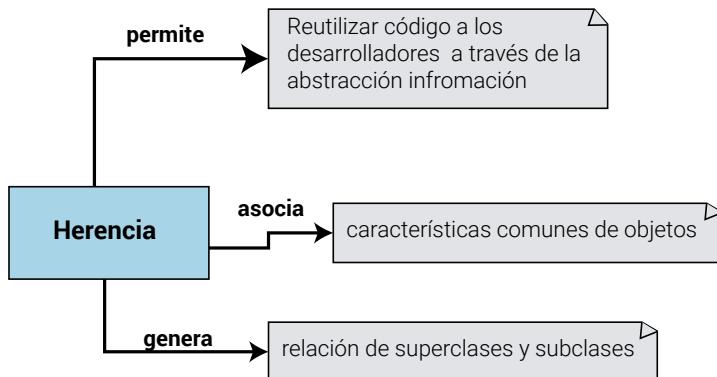


Figura 28. Concepto de Herencia.
Elaborado por: Elizalde, R. (2019)

5.1.1. Superclases y subclases

Luego de la lectura de la sección correspondiente en la guía didáctica sobre superclases y subclase, a través de la figura 29 resaltamos los puntos más importantes.

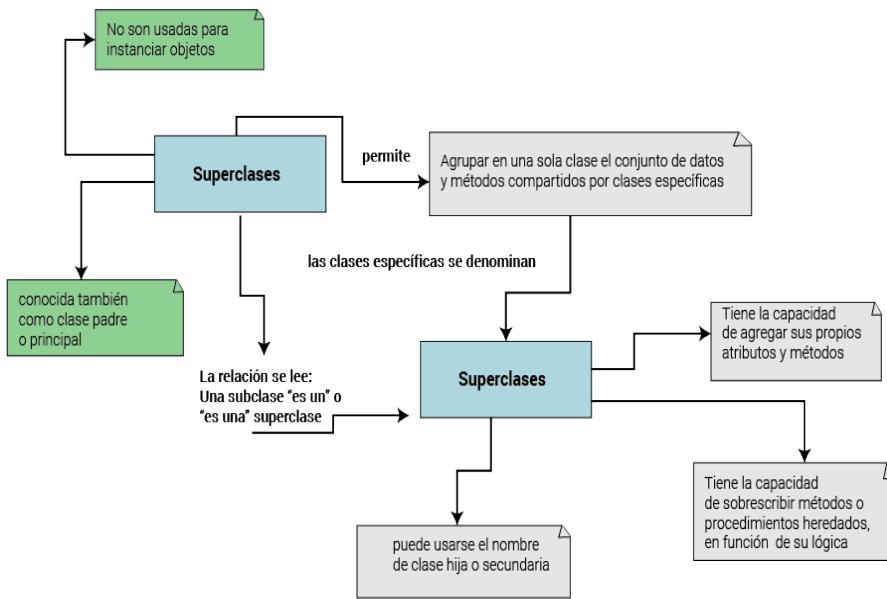


Figura 29. Características de superclases y subclases.

Elaborado por: Elizalde, R. (2019)

5.2. Uso de diagramas de clase con herencia

En la sección Uso de Diagramas de Clase con Herencia de la guía didáctica se realiza las especificaciones para el desarrollo de diagramas de clases, aplicando el concepto de herencia; por favor leer detenidamente cada acción descrita.

Se plantea el siguiente ejercicio como medio de refuerzo para el uso de diagramas de clase con herencia.

Generar una superclase llamada Tipo, con atributos: miatributo1, miatributo2; luego implementar herencia en una clase llamada TipoBasico que tiene un atributo llamado miatributo3. Expresar la solución en diagrama de clases y miniespecificación.

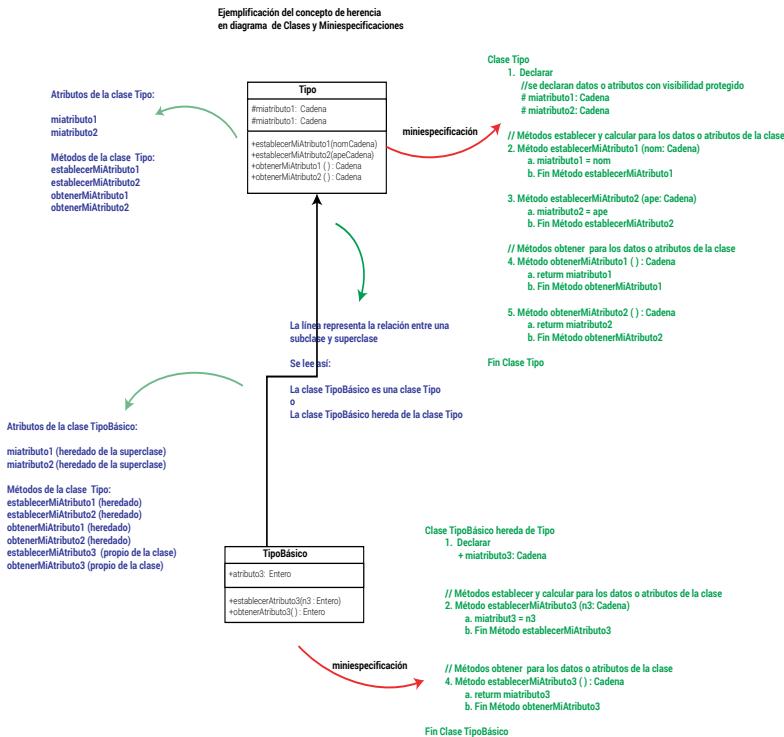


Figura 30. Solución en diagrama de clases del ejemplo planteado sobre herencia en programación orientada a objetos.

Elaborado por: Elizalde, R. (2019)

Recursos de aprendizaje

- Lea los contenidos y ejemplos de la unidad 5. Herencia en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 5 de la guía, permiten a los estudiantes conocer un concepto importante del Paradigma de Orientación a Objetos denominados Herencia, se explica su

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

implementación a través de diagramas de clase y se resalta las ventajas de uso para los desarrolladores en la solución final de diversas problemáticas planteadas. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 13. Programación Orientada a Objetos aplicando herencia; páginas 358-384

LECTURA: López, L. (2013). Metodología de la Programación Orientada a Objetos. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado con la construcción de Diagramas de Clase aplicando Herencia como parte fundamental del paradigma de Programación Orientada a Objetos; los ejercicios resueltos, permiten identificar de manera clara los elementos a considerar en el planteamiento de superclases y subclases. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Diagrama de Clases - Herencia en POO.

LECTURA: García, A., y Arranz, J. (2010). Herencia. Recuperado de http://ocw.upm.es/pluginfile.php/1037/mod_label/intro/16-herencia.pdf

El recurso educativo abierto permite exemplificar en un lenguaje de programación específico la el concepto de herencia entre clases; a través del planteamiento de problemáticas sencillas. Se aborda temas como definición de herencia y jerarquía de clases.



Semana 12

5.3. Algoritmos orientados a objetos con herencia

En la sección Algoritmos orientados a objetos con herencia de la guía didáctica se realizan las especificaciones para el desarrollo de miniespecificaciones, aplicando el concepto de herencia; por favor leer detenidamente cada acción descrita.

Aspectos por considerar para representar la herencia en miniespecificaciones.

- En la clase principal, declarar datos o atributos **visibilidad de protegido (#)**.
- En la clase secundaria o hija, para indicar que una clase hereda o se deriva de una clase se usa la forma: clase secundaria hereda de principal.
- En la clase secundaria o hija se puede:
 - Declarar datos o atributos particulares de la subclase.
 - Declarar métodos propios de la subclase.
 - (opcional) Sobrescribir métodos de la superclase.

Se retoma el ejercicio de la sección anterior del presente documento, para presentar y explicar la solución al caso presentado, pero a nivel de miniespecificaciones. Se recuerda el enunciado del problema

Generar una superclase llamada Tipo, con atributos: miatributo1, miatributo2; luego implementar herencia en una clase llamada TipoBasico que tiene un atributo llamado miatributo3. Expresar la solución en diagrama de clases y miniespecificación.

Solución en miniespecificación.

- [Enlace a los archivos de solución en miniespecificaciones.](#)

Recursos de aprendizaje

El conjunto de recursos de aprendizaje seleccionados que permiten a los estudiantes reforzar y practicar el concepto de herencia a aplicaciones de la vida real a través de la implementación de miniespecificaciones, se expone a continuación:

- Lea los contenidos y ejemplos de la unidad 5. Herencia en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 5 de la guía, permiten a los estudiantes conocer un concepto importante del Paradigma de Orientación a Objetos denominados Herencia, se explica su implementación a través del diseño de algoritmos a través de miniespecificaciones y se resalta las ventajas de uso para los desarrolladores en la solución final de diversas problemáticas planteadas. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 13 Programación Orientada a Objetos aplicando herencia; páginas 358-384

LECTURA: López, L. (2013). Metodología de la Programación Orientada a Objetos. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de miniespecificaciones aplicando Herencia como parte fundamental del paradigma de Programación Orientada a Objetos; los ejercicios resueltos, permiten identificar de manera clara los elementos a considerar en el planteamiento de pseudocódigo para exemplificar una relación entre superclase y una subclase. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Miniespecificaciones - Herencia en POO.

LECTURA: García, A., y Arranz, J. (2010). Herencia. Recuperado de http://ocw.upm.es/pluginfile.php/1037/mod_label/intro/16-herencia.pdf

El recurso educativo abierto permite exemplificar en un lenguaje de programación específico la el concepto de herencia entre clases; a través del planteamiento de problemáticas sencillas. Se aborda temas como definición de herencia y jerarquía de clases.



Actividad de aprendizaje recomendada

Actividad 1

- **Actividad de aprendizaje**

En la autoevaluación 5 se presentan preguntas relacionadas con la temática de herencia. Se solicita revisar la unidad número 5 de la presente guía y el capítulo “Programación Orientada a Objetos usando herencia” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 5.



Autoevaluación 5

A continuación, se presentan preguntas relacionadas con la temática de herencia. Se solicita revisar la unidad número 5 de la presente guía y el capítulo “Programación Orientada a Objetos usando herencia” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. En el marco del concepto de Herencia, ¿cuál de las siguientes ideas son correctas?
 - a. La herencia permite a las subclases heredar solo los datos (atributos).
 - b. La herencia permite a las subclases heredar solo los métodos de la superclase.
 - c. La herencia permite a las subclases heredar los datos (atributos) y los métodos de la superclase.

2. ¿Cuál de las siguientes frases representan la relación entre una superclase y subclase, dentro del ámbito de herencia?
 - a. Una superclase “se deriva de” subclase.
 - b. Una subclase “se deriva de” superclase.
 - c. Una subclase “tiene una” superclase

Índice

Primer bimestre

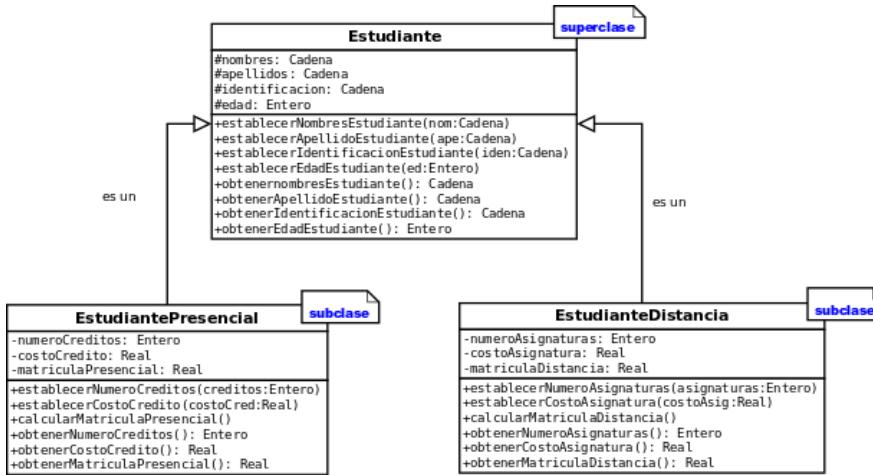
Segundo bimestre

Solucionario

Referencias bibliográficas

3. En miniespecificación, ¿cómo se declara una clase que hereda de una superclase?
 - a. Clase Herencia: Secundaria de Principal.
 - b. Clase Secundaria hereda de Principal.
 - c. Clase Secundaria tienen un Principal.
4. ¿Para qué los atributos de una superclase puedan ser manipulados por las subclases, qué tipo de visibilidad deben poseer?
 - a. Privados
 - b. Públicos
 - c. Protegidos
5. Si una subclase necesita crear sus propios datos o métodos, ¿cuál es la solución de implementación?
 - a. La subclase no puede declarar o implementar sus propios datos o métodos
 - b. La subclase puede declarar e implementar sus propios datos o métodos, dentro de su clase.
 - c. La subclase puede declarar e implementar sus propios datos o métodos, dentro de la superclase.

6. En base al siguiente diagrama de clases, determine, ¿cuál de las siguientes miniespecificaciones tiene un error de implementación?



Miniespecificaciones:

Opción A

Clase Ejecutar

Método principal()

Declarar

valor: Entero

valor = 20

EstudiantePresencial e = new EstudiantePresencial()

e.establecerNombresEstudiante(valor)

Fin Método principal

Fin Clase Ejecutar

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Opción B

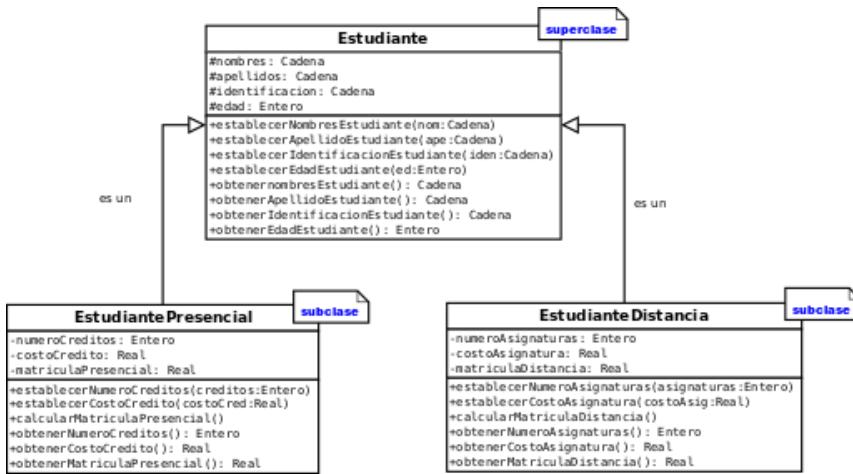
```
Clase Ejecutar
Método principal()
Declarar
valor: Cadena
valor = "José"
EstudiantePresencial e = new EstudiantePresencial()
e.establecerNombresEstudiante(valor)
Fin Método principal
Fin Clase Ejecutar
```

Opción C

```
Clase Ejecutar
Método principal()
Declarar
valor: Cadena
valor = "Sanchez"
EstudiantePresencial e = new EstudiantePresencial()
e.establecerApellidosEstudiante(valor)
Fin Método principal
Fin Clase Ejecutar
```

- a. Opción A
- b. Opción B
- c. Opción C

7. Con base en el siguiente diagrama de clases, determine la miniespecificación correcta.



Miniespecificaciones

Opción A

Clase Ejecutar

Método principal()

Declarar

valor: Entero

valor = 20

EstudiantePresencial e = new EstudiantePresencial()

e.establecerNumeroAsginaturas(valor)

Fin Método principal

Fin Clase Ejecutar

Opción B

Clase Ejecutar

Método principal()

Declarar

valor: Entero

valor = 20

```
EstudianteDistancia e = new EstudianteDistancia()
e.establecerNumeroCreditos(valor)
Fin Método principal
Fin Clase Ejecutar
```

Opción C

```
Clase Ejecutar
Método principal()
Declarar
valor: Entero
valor = 20
EstudianteDistancia e = new EstudianteDistancia()
e.establecerEdad(valor)
Fin Método principal
Fin Clase Ejecutar
```

- a. Opción A
- b. Opción B
- c. Opción C

8. Con base en la siguiente miniespecificación, determine el diagrama de clases correcto.

```
Clase Principal
Declarar
# atributo1: Cadena
# atributo2: Cadena
Método establecerAtributo1(nom: Cadena)
atributo1 = nom
Fin Método establecerAtributo1
Método establecerAtributo2(ape: Cadena)
atributo2 = ape
Fin Método establecerAtributo2
Método obtenerAtributo1() : Cadena
```

Índice

Primer
bimestre

Segundo
bimestre

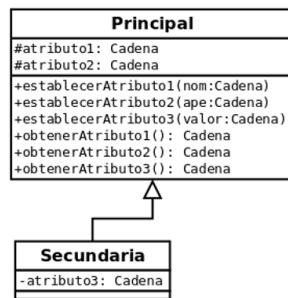
Solucionario

Referencias
bibliográficas

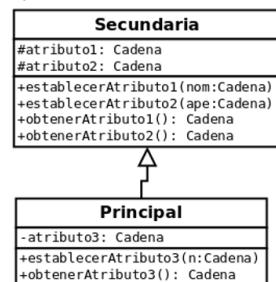
```
return atributo1
Fin Método obtenerAtributo1
Método obtenerAtributo2() : Cadena
return atributo2
Fin Método obtenerAtributo2
Fin Clase Principal
Clase Secundaria hereda de Principal
Declarar
atributo3: Cadena
Método establecerAtributo3(nom: Cadena)
atributo3 = nom
Fin Método establecerAtributo3
Método obtenerAtributo3() : Cadena
return atributo3
Fin Método obtenerAtributo3
Fin Clase Secundaria
```

Diagramas de clases

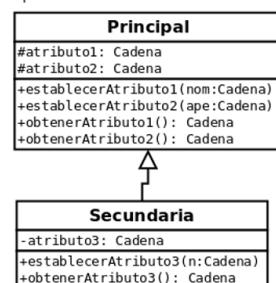
Opción A



Opción B



Opción C



- a. Opción A
- b. Opción B
- c. Opción C

9. Dadas las siguientes clases expresadas en miniespecificación:

```
Clase Principal
Declarar
# atributo1: Cadena
# atributo2: Cadena
Método establecerAtributo1(nom: Cadena)
atributo1 = nom
Fin Método establecerAtributo1
Método establecerAtributo2(ape: Cadena)
atributo2 = ape
Fin Método establecerAtributo2
Método obtenerAtributo1() : Cadena
return atributo1
Fin Método obtenerAtributo1
Método obtenerAtributo2() : Cadena
return atributo2
Fin Método obtenerAtributo2
Fin Clase Principal
Clase Secundaria hereda de Principal
Declarar
atributo3: Cadena
Método establecerAtributo3(nom: Cadena)
atributo3 = nom
Fin Método establecerAtributo3
Método obtenerAtributo3(): Cadena
return atributo3
Fin Método obtenerAtributo3
Fin Clase Secundaria
```

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Determine cuál sería la implementación correcta de una clase Ejecutar que permita presentar en pantalla:

Loja

Provincia

Ecuador

Donde:

a1 = "Loja"

a2 = "Provincia"

a3 = "Ecuador"

Opción A

Clase Ejecutar

1. Método principal()

a. Declarar Variables

a1: Cadena

a2: Cadena

a3: Cadena

b. Solicitar atributo 1,

atributo 2,

atributo 3

c. Leer a1,

a2,

a3,

d. Establecer s.establecerAtributo1(a1),

s.establecerAtributo2(a2),

s.establecerAtributo3(a3)

e. Imprimir s.obtenerAtributo1(),

s.obtenerAtributo2(),

s.obtenerAtributo3()

f. Fin Método principal

Fin Clase Ejecuta

Opción B

Clase Ejecutar

1. Método principal()
 - a. Declarar Variables
 - a1: Cadena
 - a2: Cadena
 - a3: Cadena
 - b. Secundaria s = new Secundaria()
 - c. Solicitar atributo 1,
atributo 2,
atributo 3
 - d. Leer a1,
a2,
a3,
 - e. Establecer s.establecerAtributo1(a1),
s.establecerAtributo2(a2),
s.establecerAtributo3(a3)
 - f. Imprimir s.obtenerAtributo1(),
s.obtenerAtributo2(),
s.obtenerAtributo3()
 - g. Fin Método principal

Fin Clase Ejecuta

Opción C

Clase Ejecutar

1. Método principal()
 - a. Declarar Variables
 - a1: Cadena
 - a2: Cadena
 - a3: Cadena
 - b. Secundaria s = new Secundaria()
 - c. Solicitar atributo 1,
atributo 2,
atributo 3

- d. Leer a1,
a2,
a3,
 - e. Imprimir s.obtenerAtributo1(),
s.obtenerAtributo2(),
s.obtenerAtributo3()
 - f. Fin Método principal
- Fin Clase Ejecuta
- a. Opción A
 - b. Opción B
 - c. Opción C
10. Dadas las siguientes clases expresadas en miniespecificación:

Clase Principal

- 1. Declarar
 - # atributo1: Cadena
 - # atributo2: Cadena
 - 2. Método establecerAtributo1(nom: Cadena)
 - a. atributo1 = nom
 - b. Fin Método establecerAtributo1
 - 3. Método establecerAtributo2(ape: Cadena)
 - a. atributo2 = ape
 - b. Fin Método establecerAtributo2
 - 4. Método obtenerAtributo1(): Cadena
 - a. return atributo1
 - b. Fin Método obtenerAtributo1
 - 5. Método obtenerAtributo2(): Cadena
 - a. return atributo2
 - b. Fin Método obtenerAtributo2
- Fin Clase Principal
- Clase Secundaria hereda de Principal

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

1. Declarar
atributo3: Cadena
2. Método establecerAtributo3(nom: Cadena)
a. atributo3 = nom
b. Fin Método establecerAtributo3
3. Método obtenerAtributo3(): Cadena
a. return atributo3
b. Fin Método obtenerAtributo3
Fin Clase Secundaria
Clase Secundaria2 hereda de Principal
1. Declarar
atributo4: Cadena
2. Método establecerAtributo4(nom: Cadena)
a. atributo4 = nom
b. Fin Método establecerAtributo4
3. Método obtenerAtributo4(): Cadena
a. return atributo4
b. Fin Método obtenerAtributo4
Fin Clase Secundaria2
Determine cuál de las siguientes implementaciones de la clase Ejecuta, no generaría ningún error.

Opción A

Clase Ejecutar
1. Método principal()
a. Declarar Variables
a1: Cadena
a2: Cadena
a3: Cadena
b. Secundaria s = new Secundaria()
c. Solicitar atributo 1, atributo 2, atributo 3
d. Leer a1, a2, a3,
e. Establecer s.establecerAtributo1(a1),

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

s.establecerAtributo2(a2),
s.establecerAtributo4(a3)
f. Imprimir s.obtenerAtributo1(),
s.obtenerAtributo2(),
s.obtenerAtributo4()
g. Fin Método principal
Fin Clase Ejecuta

Opción B

Clase Ejecutar
1. Método principal()
a. Declarar Variables
a1: Cadena
a2: Cadena
a3: Cadena
b. Secundaria s = new Secundaria()
c. Solicitar atributo 1, atributo 2, atributo 3
d. Leer a1, a2, a3,
e. Establecer s.establecerAtributo1(a1),
s.establecerAtributo2(a2),
s.establecerAtributo3(a3)
f. Imprimir s.obtenerAtributo1(),
s.obtenerAtributo2(),
s.obtenerAtributo3()
g. Fin Método principal
Fin Clase Ejecuta

Opción C

Clase Ejecutar
1. Método principal()
a. Declarar Variables
a1: Cadena
a2: Cadena

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

a3: Cadena

- b. Secundaria2 s = new Secundaria2()
- c. Solicitar atributo 1, atributo 2, atributo 3
- d. Leer a1, a2, a3,
- e. Establecer s.establecerAtributo1(a1),
s.establecerAtributo2(a2),
s.establecerAtributo3(a3)
- f. Imprimir s.obtenerAtributo1(),
s.obtenerAtributo2(),
s.obtenerAtributo4()
- g. Fin Método principal
Fin Clase Ejecuta

- a. Opción A
- b. Opción B
- c. Opción C

Ir al solucionario

Buen trabajo. Ahora es tiempo de revisar la unidad relacionada con Polimorfismo. ¡Avancemos!

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultados de aprendizaje 6, 7, 8 y 9

- Diseña, implementa, prueba y depura la implementación de una relación entre objetos usando una jerarquía herencia de clases.
- Discute e identifica los conceptos de encapsulación, abstracción, herencia y polimorfismo.
- Diseña, implementa y prueba la implementación de una relación “is-a” entre objetos usando una jerarquía herencia de clases.
- Compara y contrastar las nociones de los métodos de overloading and overriding en un lenguaje orientado a objetos.

Contenidos, recursos y actividades de aprendizaje

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Semana 13

Es importante indicar que los estudiantes requieren poner énfasis en el estudio a nivel teórico y práctico, sobre las directrices necesarias para implementar el polimorfismo en una problemática específica; se dará importancia al desarrollo de diagramas de clase y miniespecificaciones; sentando las bases para que el estudiante en lo posterior pueda implementar soluciones en lenguajes de alto nivel.

Luego de entender el concepto de herencia, es momento de estudiar el proceso de implementación denominado polimorfismo a través del uso de clases y métodos abstractos. En la semana 5 de actividades se procede a explicar la forma de implementación de polimorfismo mediante diagramas de clases, basado en diversas problemáticas planteadas.



Unidad 6. Polimorfismo en Programación Orientada a Objetos

Estimado estudiante: hemos llegado a la última unidad de Programación Orientada a Objetos, que satisfacción saber que su esfuerzo y dedicación les han permitido llegar hasta este punto. Para el estudio del presente apartado, se solicita revisar el contenido de:

- Texto básico en el capítulo “Programación orientada a objetos usando polimorfismo”, secciones: Polimorfismo, Diseño del diagrama de clases con polimorfismo y Diseño de algoritmos OO usando polimorfismo.
- Guía didáctica unidad 6 “Polimorfismo en Programación Orientada a Objetos” secciones Concepto de polimorfismo - Clases y métodos abstractos y Uso de Diagramas de clase con polimorfismo.

6.1. Concepto de polimorfismo

En relación con el contenido de la guía didáctica en la sección Concepto de polimorfismo se genera la figura 31, como refuerzo para el entendimiento del nuevo concepto.

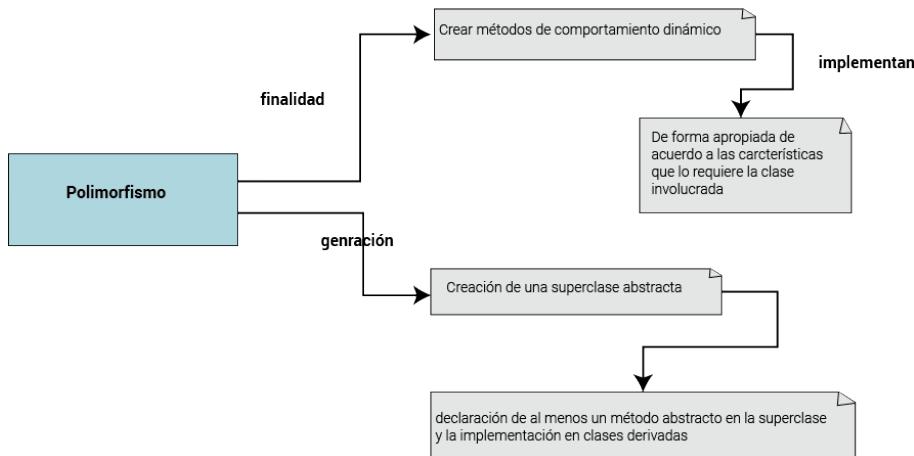


Figura 31. Concepto de Polimorfismo.

Elaborado por: Elizalde, R. (2019)

6.1.1. Clases y métodos abstractos

En la guía didáctica, sección Clases y métodos abstractos, se describen ideas relacionadas a las clases y métodos abstractos, a través de la figura 32 representamos dichos conceptos.

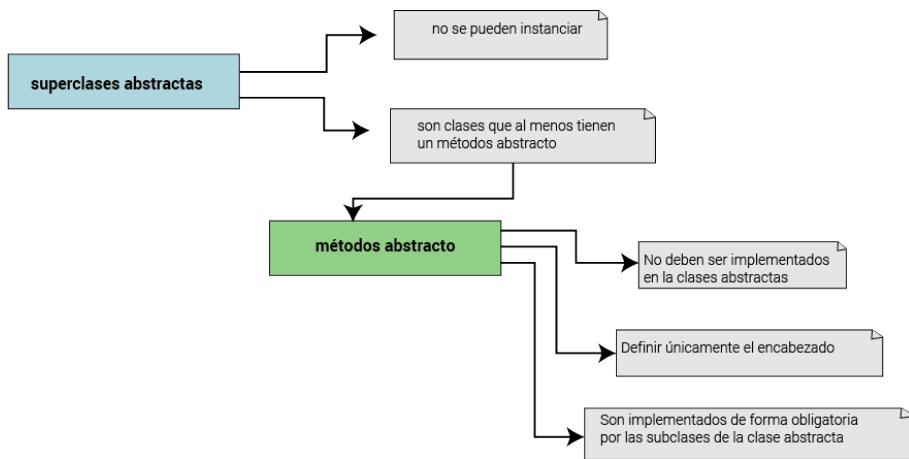


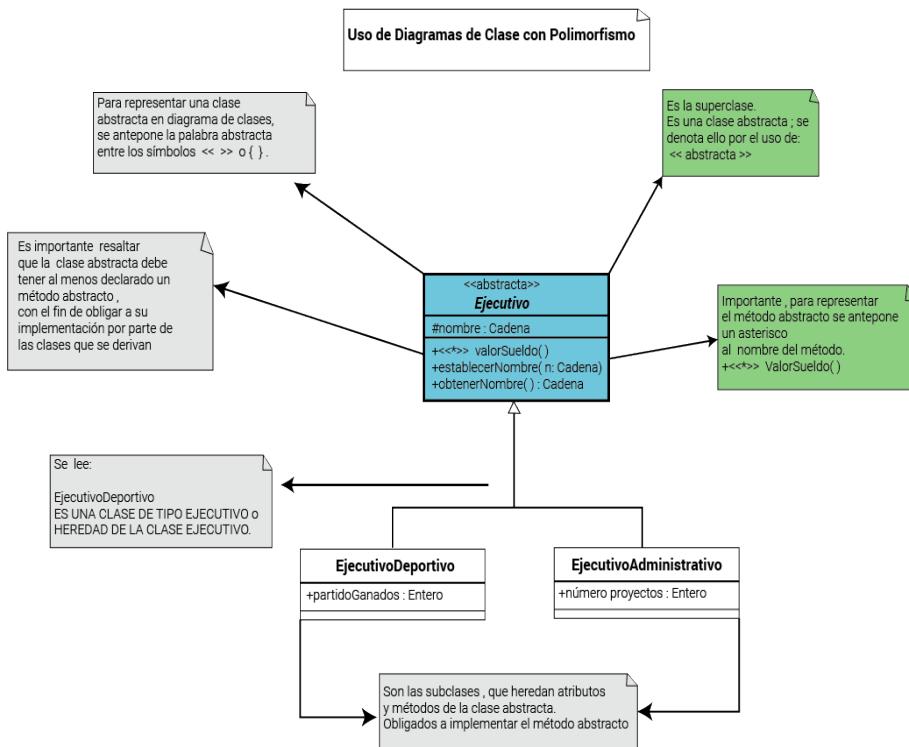
Figura 32. Clases abstractas y métodos abstractos.

Elaborado por: Elizalde, R. (2019)

6.2. Uso de diagramas de clase con polimorfismo

En la guía didáctica, sección Uso de diagramas de clase con polimorfismo, se indican las figuras y conectores que se deben usar para la representación correcta del polimorfismo a través de diagramas; además, se describe un ejemplo para su compresión.

A continuación, a través de la figura 33 describimos algunas pautas importantes para la generación de diagramas de clases aplicando polimorfismo.



*Figura 33. Uso de diagramas de clases con polimorfismo.
Elaborado por: Elizalde, R. (2019)*

Recursos de aprendizaje

Los recursos seleccionados para profundizar el aprendizaje de la semana de estudios, se los lista a continuación:

- Lea los contenidos y ejemplos de la unidad 6. Polimorfismo en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 6 de la guía, permiten a los estudiantes conocer un concepto importante del Paradigma de Orientación a Objetos denominados polimorfismo, se explica su implementación a través de diagramas de clase y se resalta las ventajas de uso para los desarrolladores en la solución final de diversas problemáticas planteadas. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 14. Programación Orientada a Objetos aplicando polimorfismo; páginas 386 - 409

LECTURA: López, L. (2013). Metodología de la Programación Orientada a Objetos. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado con la construcción de diagramas de clase aplicando polimorfismo como parte fundamental del paradigma de Programación Orientada a Objetos; los ejercicios resueltos, permiten identificar de manera clara los elementos a considerar en el planteamiento de clases y métodos abstractos. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Diagramas de clases-Polimorfismo en POO.

LECTURA: Pereira, M. (2015). Interfaces y polimorfismo. Recuperado de http://ocw.uc3m.es/historico/programacion-java/transparencias/tema10.pdf/at_download/file



Semana 14

Luego de entender el concepto de herencia, es momento de estudiar el proceso de implementación denominado polimorfismo, a través del uso de clases y métodos abstractos. En la semana 6 de actividades, se procede a explicar la forma de implementación de polimorfismo a través de miniespecificaciones, mediante diversas problemáticas planteadas.

6.3. Algoritmos orientados a objetos con polimorfismo

Continuamos con el estudio de la unidad 6, relacionada con la temática de polimorfismo en Programación Orientada a Objetos. La semana anterior analizamos los procedimientos para implementar y representar polimorfismo a través diagramas de clases.

Es momento de entender los pasos a seguir para desarrollar **miniespecificaciones** bajo el concepto de polimorfismo; de igual manera, revisar el uso del lenguaje de programación Java para la exemplificación.

Con base en lo descrito en la guía didáctica, sección Algoritmos orientados a objetos con polimorfismo, se necesita considerar algunas pautas iniciales:

- Declarar como superclase una clase de tipo abstracta.
- En la declaración de la clase se agrega la palabra abstracta, de la forma: Clase abstracta Superclase.
- La superclase abstracta debe tener al menos un método abstracto sin implementar (la tarea de implementar dicho método, se lo dejamos a las clases hijas de la superclase). Un método abstracto se lo declara de la siguiente forma: Método abstracto calcularSueldo().

En la guía didáctica se describe un ejemplo completo (diagrama de clases, miniespecificación e implementación en lenguaje de programación).

Se solicita revisar los siguientes enlaces, con el objetivo que pueda analizar y ejecutar el programa en su computadora personal.

Diagrama

[Desarrollo del caso planteado mediante diagrama.](#)

Miniespecificación

[Desarrollo del caso planteado mediante miniespecificación.](#)

Lenguaje de programación

[Desarrollo del caso planteado mediante lenguaje de programación.](#)

Para afianzar los conceptos relacionados con el polimorfismo, planteamos el siguiente ejercicio. Problema: exemplificar el concepto de polimorfismo entre las siguientes clases.

- Superclase Abstracta DeportistaAltoRendimiento: atributos: #nombre: Cadena y #incentivoMensual: Real; método abstracto: calcularIncentivo.

- Clases que heredan de la superclase:
 - JugadorFutbol: atributos (numeroPartidosJugados, numeroGoles).
 - Atleta: atributo (numeroCompetencias).

Para la exemplificación, usar una clase llamada EjecutaPolimorfismo como principal.

A continuación se listan los enlaces, donde encontrará la solución propuesta.

- Solución en miniespecificación: [enlace a los archivos en miniespecificaciones](#)
- Solución en lenguaje de programación: [enlace a los archivos en lenguaje de programación Java](#).

Recursos de aprendizaje

- Lea los contenidos y ejemplos de la unidad 6. Polimorfismo en Programación Orientada a Objetos.

LECTURA: Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Los contenidos de la unidad 6 de la guía, permiten conocer un concepto importante del Paradigma de Orientación a Objetos denominados polimorfismo, se explica su implementación a través del diseño de algoritmos a través de miniespecificaciones y se resalta las ventajas de uso para los desarrolladores en la solución final de diversas

problemáticas planteadas. Luego de la lectura y comprensión de los conceptos y ejercicios de la unidad, se recomienda al estudiante plantearse sus propios ejercicios.

- Revise los conceptos y ejercicios de la unidad 14 Programación Orientada a Objetos aplicando polimorfismo; páginas 386 - 409

LECTURA: López, L. (2013). Metodología de la Programación Orientada a Objetos. Ciudad de México, México: Alfaomega.

La lectura permite al estudiante comprender lo relacionado al diseño de miniespecificaciones aplicando Polimorfismo como parte fundamental del paradigma de Programación Orientada a Objetos; los ejercicios resueltos, permiten identificar de manera clara los elementos a considerar en el planteamiento de pseudocódigo para exemplificar clases y métodos abstractos. Se recomienda usar organizadores de estudio gráficos para obtener mejores resultados de las lecturas anteriormente descritas.

- Considere el recurso colocado en el entorno virtual de aprendizaje sobre: Algoritmos Orientados a Objetos con Polimorfismo.

LECTURA: Pereira, M. (2015). Interfaces y polimorfismo. Recuperado de <http://ocw.uc3m.es/cursos-archivados/programacion-java/transparencias/tema10.pdf>

El recurso educativo abierto permite exemplificar en un lenguaje de programación específico la el concepto de polimorfismo entre clases; a través del planteamiento de problemáticas sencillas. Se aborda temas como definición y características del polimorfismo.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Actividad de aprendizaje recomendada

Actividad 1

- **Actividad de aprendizaje**

En la autoevaluación 6 se presentan preguntas relacionadas con la temática polimorfismo. Se solicita revisar la unidad número 6 de la presente guía y el capítulo “Programación Orientada a Objetos usando polimorfismo” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación. Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 6.

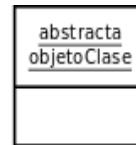


Autoevaluación 6

A continuación, se presentan preguntas relacionadas con la temática de polimorfismo. Se solicita revisar la unidad número 6 de la presente guía y el capítulo "Programación Orientada a Objetos usando polimorfismo" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

Los siguientes enunciados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

1. ¿ Cuál es la forma correcta de representar una clase abstracta en un diagrama de clases?



- a. A
- b. B
- c. C

2. ¿ Cuál es la característica de una clase abstracta?

- a. Es opcional poseer un método abstracto.
- b. Tener al menos un método abstracto.
- c. Implementar al menos un método abstracto.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

3. Se necesita declarar un método abstracto en un diagrama de clases ¿ Cuál es la opción correcta?
- #<<*>> método_abstracto().
 - +<<*>> método_abstracto().
 - + método_abstracto().
4. Se tiene las siguientes miniespecificaciones, determina la opción correcta para la implementación de una clase abstracta.

Opción A

Clase abstracta Estudiante

1. Declarar
 2. Método abstracto calcularMatricula()
// método sin implementar
- Fin Clase Estudiante

Opción B

Clase Estudiante

1. Declarar
 2. Método abstracto calcularMatricula()
// método sin implementar
- Fin Clase Estudiante

Opción C

Clase abstracta Estudiante

1. Declarar
 - a: Entero
b: Entero
c: Entero
 2. Método abstracto calcularMatricula()
 $c = a + b$
- Fin Clase Estudiante

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- a. Opción A
 - b. Opción B
 - c. Opción C
5. ¿La visibilidad de los datos de una superclase abstracta, de qué tipo deben ser?
- a. Privados
 - b. Públicos y privados
 - c. Protegidos
6. Se necesita implementar polimorfismo entre una superclase y subclase. ¿Cuál es la opción correcta entre las siguientes miniespecificaciones?

Opción A

Clase abstracta Estudiante

1. Declarar
 - a: Entero
 - b: Entero
2. Método abstracto calcularMatricula()
// método sin implementar
Fin Método calcularMatricula
- Fin Clase Estudiante

Clase Alumno

1. Declarar
2. Método calcularMatricula()
 $c = a * b$
Fin Método calcularMatricula
- Fin Clase Alumno

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Opción B

```
Clase abstracta Estudiante
1. Declarar
a: Entero
b: Entero
2. Método abstracto calcularMatricula()
// método sin implementar
Fin Método calcularMatricula
Fin Clase Estudiante
Clase Alumno hereda de Estudiante
1. Declarar
2. Método abstracto calcularMatricula()
c = a*b
Fin Método calcularMatricula
Fin Clase Alumno
```

Opción C

```
Clase abstracta Estudiante
1. Declarar
a: Entero
b: Entero
2. Método abstracto calcularMatricula()
// método sin implementar
Fin Método calcularMatricula
Fin Clase Estudiante
Clase Alumno hereda de Estudiante
1. Declarar
2. Método calcularMatricula()
c = a*b
Fin Método calcularMatricula
Fin Clase Alumno
```

- a. Opción A
 - b. Opción B
 - c. Opción C
7. ¿Cuál es la finalidad de la implementación del polimorfismo?
- a. El polimorfismo se usa con la finalidad de crear métodos de comportamiento estáticos que se implementan de forma apropiada de acuerdo a las características que lo requiere la clase involucrada.
 - b. El polimorfismo se usa con la finalidad de crear métodos de comportamiento dinámico que se implementan de forma apropiada de acuerdo a las características que lo requiere la clase involucrada.
 - c. El polimorfismo se usa con la finalidad de crear métodos de comportamiento dinámico que se implementan solo en las llamadas superclases.
8. En la siguiente miniespecificación existe un error, se solicita identificarlo.

Clase abstracta Estudiante

1. Declarar

a: Entero

b: Entero

2. Método abstracto calcularMatricula()

// método sin implementar

Fin Método calcularMatricula

Fin Clase Estudiante

Clase EjecutaEstudiante

1. Declarar

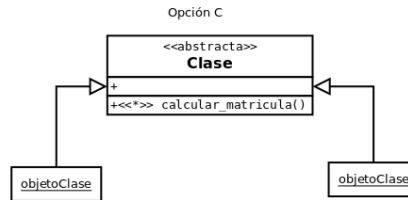
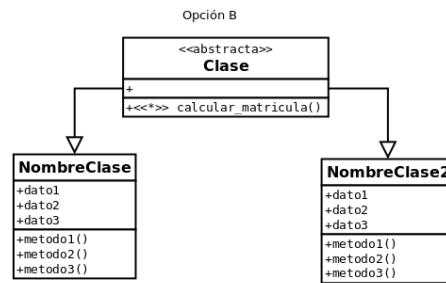
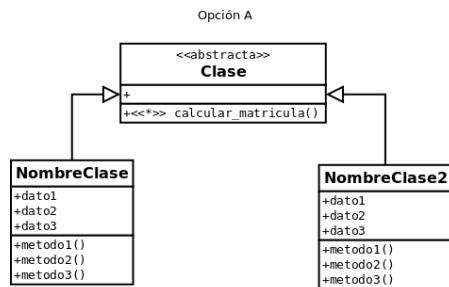
2. Método principal()

Estudiante e = new Estudiante()

Fin Método principal

Fin Clase Alumno

- a. En la clase abstracta Estudiante están mal declarados los datos o atributos.
 - b. La clase EjecutaEstudiante debe heredar de la clase Estudiante.
 - c. En el método principal de la clase EjecutaEstudiante, se está creando un objeto de una clase abstracta.
9. Con base en los siguientes diagramas de clases, determine cuál es el correcto en relación con los conceptos de polimorfismo estudiados.



- a. Opción A
- b. Opción B
- c. Opción C

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

10. Dada la siguiente miniespecificación, donde se declara una clase abstracta:

```
Clase abstracta Estudiante  
Declarar  
a: Entero  
b: Entero  
c: Entero  
Método abstracto promedio()  
// método sin implementar  
Fin Método calcularMatricula  
Fin Clase Estudiante
```

Determine, ¿cuál de las siguientes clases, se puede considerar como una clase derivada o subclase de la superclase Estudiante?

Opción A

```
Clase Estudiante1 hereda de Estudiante  
Declarar  
Método promedio( )  
c = a * b  
Fin Método promedio  
Fin Clase Estudiante1
```

Opción B

```
Clase Estudiante1  
Declarar  
Método promedio( )  
c = a * b  
Fin Método promedio  
Fin Clase Estudiante1
```

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Opción C

```
Clase Estudiante1 hereda de Estudiante
Declarar
    Método promedio_estudiante()
    c = a * b
Fin Método promedio
Fin Clase Estudiante1
```

- a. Opción A
- b. Opción B
- c. Opción C

[Ir al solucionario](#)

Hemos concluido con éxito el estudio de nuestra sexta unidad y con ello la guía didáctica.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Actividades finales del bimestre



Semana 15



Actividad de aprendizaje recomendada

Actividad 1:

- **Actividad de aprendizaje**

La autoevaluación 4 se plantea a través de preguntas relacionadas al manejo de arreglos en Programación Orientada a Objetos. Se solicita revisar la unidad 4 de la presente guía y el capítulo "Programación Orientada a Objetos aplicando arreglos" del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 4.

Actividad 2

- **Actividad de aprendizaje**

La autoevaluación 5 presenta preguntas relacionadas con la temática de herencia. Se solicita revisar la unidad número 5 de la presente guía y el capítulo “Programación Orientada a Objetos usando herencia del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 5.

Actividad 3

- **Actividad de aprendizaje**

En la autoevaluación 6 se presentan preguntas relacionadas con la temática de polimorfismo. Se solicita revisar la unidad número 6 de la presente guía y el capítulo “Programación Orientada a Objetos usando polimorfismo” del texto básico para contestar las interrogantes. La autoevaluación le permitirá reafirmar los conceptos estudiados.

- **Procedimiento**

Los enunciados planteados son de opción múltiple, cada uno de ellos tiene una sola respuesta correcta. Lea atentamente cada

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

pregunta y seleccione la opción que usted considere como válida en cada situación.

Luego de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica, en lo relacionado con la autoevaluación 6



Semana 16

Actividad 1

La semana 8 se define como un espacio que permite a los estudiantes reforzar temáticas no comprendidas en las primeras semanas de estudio, de cara a la evaluación presencial del segundo bimestre; por ello, se propone el repaso de las unidades de estudio revisadas hasta el momento:

- Unidad 4. Estructuras de datos en Programación Orientada a Objetos.
- Unidad 5. Herencia en Programación Orientada a Objetos.
- Unidad 6. Polimorfismo en Programación Orientada a Objetos.

La revisión consiste en realizar las siguientes acciones:

- Revisar los recursos de aprendizaje recomendados en cada semana.
- Realizar nuevamente las autoevaluaciones de cada unidad del segundo bimestre.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

- Revisar las preguntas de los cuestionarios de refuerzo.
- Plantearse ejercicios y resolverlos.

A partir de ahora usted está preparado para continuar con las nuevas actividades académicas necesarias para lograr su formación profesional.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

4. Solucionario

Primer bimestre

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1.	b	Para solucionar problemáticas a través del paradigma de Orientación a Objetos, se debe representar la solución como una interacción de objetos, que permiten realizar entrada de datos, procesamiento de información y salida de datos para una solución propuesta. Por lo tanto, la opción correcta es la b.
2.	b	En la sección 8.1.1 del texto básico, se menciona que un objeto es todo lo que involucra el dominio del problema planteado; es decir, son entidades o sujetos. Por lo tanto, la opción correcta es la b.
3.	a	En el punto 1.2.1 de la guía, se menciona que en el paradigma de orientación a objetos se estable que los objetos están formados por elementos (datos y métodos). Por lo tanto, la opción correcta es la a.
4.	b	En el punto 1.2.1 de la guía, se manifiesta que en otros paradigmas de programación los métodos tienen otras denominaciones como: procedimientos, módulos, funciones, subrutinas, entre otros. Por lo tanto, la opción correcta es la b.
5.	b	Para representar un objeto en diagrama, se usa un rectángulo, en la parte superior se ubica el nombre de objeto subrayado y en la parte media, se ubica el conjunto de datos o atributos. Por lo tanto, la opción correcta es la b.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
6.	a	El enunciado solicita la representación de una colección de objetos con los atributos (datos): nombres, apellidos, identificación y número de créditos. La opción A, representa dos objetos (objetoEstudiante1 y objetoEstudiante2) y los mismos contienen información para todos los datos solicitados por el enunciado. La opción B y C, no cumplen con los requerimientos solicitados (falta de datos). Por lo tanto, la opción correcta es la a.
7.	a	En la sección 1.2.2 de la guía se indica que las clases se caracterizan por tener los mismos datos y métodos. Por lo tanto, la opción correcta es la a.
8.	a	La representación a través de un diagrama, de una instancia y sus objetos se realiza haciendo uso de una línea discontinua que denote la relación "es una instancia de". La opción A representa correctamente una clase llamada Estudiante y tres objetos (objetoEstudiante1, objetoEstudiante2, objetoEstudiante3) que se comunican con la clase Estudiante, mediante una línea discontinua. Por lo tanto, la opción correcta es la a.
9.	b	En la opción B, existe representada una clase que tiene las siguientes características: dos atributos, un atributo privado (- dato1) y un atributo protegido (# dato2); tres métodos, un método público (+ metodo1()), dos métodos protegidos (# metodo2(), # método3()). Por lo tanto, cumple con lo requerido por el enunciado. La respuesta correcta es la b.
10.	b	En la sección 1.5 se manifiesta lo siguiente: la arquitectura modelo-vista-controlador se usa en el desarrollo de soluciones bajo el paradigma de programación orientado a objetos. Por lo tanto, la opción correcta es la b.

Ir a la
autoevaluación



[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
1.	a	<p>En la sección 2.1 de la guía, se indica que la forma para iniciar un miniespecificación es a través de: Algoritmo NOMBRE_DEL_ALGORITMO Por lo tanto, la opción correcta es la a.</p>
2.	a	<p>En el punto 2.1 de la guía, se indica que la característica de los métodos establecer (set) es la de colocar el valor que se desea asignar al dato (atributo o variable). Por lo tanto, la opción correcta es la a.</p>
3.	a	<p>En el punto 2.1 de la guía, se indica que la característica de los métodos obtener (get) es poder acceder a un valor de un dato (variable). Por lo tanto, la opción correcta es la a.</p>
4.	b	<p>En el punto 2.1 de la guía, se indica que la forma de declarar e inicializar un objeto de una clase es la siguiente: ClaseDemostrativa objeto = new ClaseDemostrativa() Por lo tanto, la opción correcta es la b.</p>
5.	a	<p>Con base en la clase declarada, se puede manifestar: La opción A declara e inicializa de manera correcta la clase Trabajador; y a través del objeto creado t, llama al método establecerSueldo de manera correcta, pues le está enviando un parámetro tipo real, como lo requiere el método declarado en la clase Trabajador. La opción B, no inicializa de manera correcta la clase Trabajador, falta la palabra reservada new. La opción C, declara e inicializa de manera correcta la clase Trabajador; y a través del objeto creado t, llama al método obtenerSueldo de manera incorrecta, pues le está enviando un parámetro tipo real, pero el método declarado en la clase Trabajador, no recibe ningún parámetro. Por lo tanto, la opción que no genera errores es la a.</p>
6.	b	<p>En la sección 2.2 de la guía se manifiesta que todas las clases tienen por defecto un constructor, mismo que es llamado al momento de crear la instancia del objeto. Por lo tanto, la opción correcta es la b.</p>

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
7.	a	<p>Con base en las clases Vehiculo y Ejecuta se imprimirá por pantalla lo siguiente:</p> <ul style="list-style-type: none">▪ La sentencia Vehiculo v = new Vehiculo(), crea e inicializa un objeto v de tipo Vehiculo, al momento de inicializar se llama al constructor declarado, en el cual se inicializa la el atributo placa con el valor LBBC-0183. El paso b del método principal de la clase Ejecuta, se llama al método obtenerPlaca(), el mismo retorna el valor que tenga asignado en ese momento la variable placa (para el caso presentado es LBBC-0183).▪ La sentencia v.establecerPlaca("PBCC-0192"), llama al método establecerPlaca y le envía un valor, el método recibe el valor y lo asigna a la variable placa, reemplazando el valor que haya tenido hasta ese momento. El paso d del método principal de la clase Ejecuta, se llama al método obtenerPlaca(), el mismo retorna el valor que tenga asignado en ese momento la variable placa (para el caso presentado es PBCC-0192). <p>Por lo tanto, la opción correcta es la a.</p>
8.	c	<p>La clase Vehiculo, posee dos constructores:</p> <ul style="list-style-type: none">▪ Un constructor que no recibe parámetros, Método Vehiculo()▪ Un constructor que recibe un parámetro tipo cadena. Método Vehiculo(pl: Cadena) <p>Con base en lo anterior y tomando en consideración que en la opción C, se llama un constructor enviando dos parámetros tipo cadena Vehiculo v = new Vehiculo("ADEA-1234", "TGBA-4321"), se puede inferir que sucederá un error, ya que no existe un constructor en la clase Vehiculo que requiera dos parámetros tipo cadena. Por lo tanto, la opción correcta es la c.</p>
9.	a	<p>En la clase Ejecuta en el paso b, se solicita un valor para la variable opción, si el valor de opción es 1, se declara e inicializa un objeto tipo Vehículo haciendo uso del constructor que recibe un parámetro. Por lo tanto, la opción correcta es la a.</p>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
10.	a	En la sección 2.2 de la guía, se manifiesta que en la declaración del método constructor de una clase se usa el mismo nombre de la clase seguido de un paréntesis (). Por lo tanto, la opción correcta es la a.

Ir a la
autoevaluación

Autoevaluación 3

Pregunta	Respuesta	Retroalimentación
1.	c	<p>La opción C permite ejecutar un proceso repetitivo en 5 oportunidades, desde contador = 1 hasta que contador sea ≤ 5, y en cada iteración realiza:</p> <ul style="list-style-type: none"> ▪ Crea e inicializa un objeto llamado d de tipo Docente. ▪ Solicita ingresar un valor para la variable e. ▪ Se usa el método establecerEdad enviando como parámetro el valor previamente ingresado en la variable e. <p>Por lo tanto, la respuesta correcta es la c.</p>
2.	b	<p>Con base en las clases Docente y Ejecutar, se puede afirmar que a través del ciclo do-while se puede crear al menos un objeto tipo Docente; pues en la primera iteración del ciclo repetitivo la variable contador se incrementa en 1, quedando la comparación de final de ciclo $2 < 1$, al ser falsa la comparación, el ciclo repetitivo termina.</p>
3.	c	<ul style="list-style-type: none"> ▪ La opción A tiene un ciclo repetitivo for de la siguiente manera: for i=5; i<5;i++; en el mismo se puede apreciar que no permitirá iterar en ninguna oportunidad, pues el contador i tiene un valor de 5 y la comparación $i < 5$ sería falsa. ▪ La opción B tiene un ciclo repetitivo while de la siguiente manera: while contador <5; en el mismo se puede apreciar que no permitirá iterar en ninguna oportunidad, pues la variable contador tiene un valor de 5 (contador = 5) y la comparación $contador < 5$ sería falsa.
		<ul style="list-style-type: none"> ▪ La opción C tiene un ciclo repetitivo do-while de la siguiente manera (comparación al final del ciclo): while contador <5; en el mismo se puede apreciar que se permitirá iterar varias veces, pues la variable contador tiene un valor inicial de 1 (contador = 1) y la comparación $contador < 5$ sería verdadera, hasta que el contador vaya incrementando de 1 en 1 (con base en la sentencia contador = contador + 1) y alcance el valor de 5. <p>Por lo tanto, la opción correcta es la c.</p>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 3		
Pregunta	Respuesta	Retroalimentación
4.	b	<ul style="list-style-type: none">▪ La opción A permite iterar en 5 oportunidades, en cada iteración permite crear e inicializar un objeto llamado d, de tipo Trabajador; luego solicita el ingreso de un valor para la variable e; posteriormente, a través del objeto d llama al método obtenerSueldo y le envía como parámetro la variable e; esto genera un error debido a que el método obtenerSueldo, no recibe ningún parámetro.▪ La opción B permite iterar en 5 oportunidades, en cada iteración permite crear e inicializar un objeto llamado d, de tipo Trabajador; luego solicita el ingreso de un valor para la variable e; posteriormente, a través del objeto d llama al método establecerSueldo y le envía como parámetro la variable e; a su vez el método recibe el parámetro y asigna el mismo al atributo sueldo de la clase Trabajador.▪ La opción C permite iterar en 5 oportunidades, en cada iteración solo permite crear e inicializar un objeto llamado d, de tipo Trabajador. <p>Por lo tanto, la opción correcta es la b.</p>
5.	c	Con base en las sentencias: while contador <=3 y while contador <3 , la miniespecificación permitirá crear tres objetos de tipo Trabajador, si el valor ingresado para la variable opción es igual a 1; en caso de ingresar un valor diferente de 1 para la variable opción, se generará un proceso que permite crear dos objetos de tipo Trabajador.

Ir a la
autoevaluación



Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Segundo bimestre

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
1.	a	<p>En la sección 4.1 de la guía, se manifiesta que en la declaración de atributos o datos de tipo arreglo se usa la palabra reservada Arreglo, el tamaño del arreglo entre [] y el tipo de dato a usar para el arreglo.</p> <p>Por lo tanto, la opción correcta es la a.</p>
2.	c	<p>En la sección 4.1 de la guía, se manifiesta que en la declaración de atributos o datos de tipo arreglo se usa la palabra reservada Arreglo, el tamaño del arreglo entre [] y el tipo de dato a usar para el arreglo.</p> <p>Por lo tanto, la opción correcta es la a.</p>
3.	a	<p>La opción correcta que permite generar un método establecer y un método obtener para el atributo promedios de la clase Clase, es la a; pues en dicha opción el método establecerPromedios recibe un parámetro tipo arreglo de enteros y lo asigna a una variable llamada promedios (dicha variable está declarada así: <<Arreglo>>[12] Entero); además el método obtenerPromedios, retorna el valor que tendría asignado en un determinado instante el atributo promedios.</p>
4.	a	<p>La opción correcta es la a, pues en dicha opción se genera un método que permite acumular en una variable suma, cada uno de los valores que tenga el atributo tipo arreglo llamado valores (se hace uso de un ciclo repetitivo, for i = 0; i<3; i++), a través de la sentencia suma = suma + valores[i]</p>
5.	a	<p>El método establecerValores recibe como parámetro un arreglo de datos de tipo Docente: establecerValores(listaValores[]: <<Arreglo>> Docente).</p> <p>La opción correcta es la a, pues al hacer uso del método establecerValores, envía una variable llamada lista_valores, que es un arreglo de tipo Docente.</p>

Ir a la
autoevaluación



Autoevaluación 5		
Pregunta	Respuesta	Retroalimentación
1.	c	<p>En la sección 5.1.1 de la guía se menciona que la herencia permite asociar en una sola clase (superclase) el conjunto de datos o atributos y métodos que sean compartidos por un conjunto de clases específicas (subclases). Por lo tanto, la opción correcta es la c.</p>
2.	b	<p>La relación entre las subclases y superclases se representa con las siguientes frases:</p> <ul style="list-style-type: none"> ▪ Una subclase "es un" o "es una" superclase ▪ Una subclase "se deriva de" superclase <p>Por lo tanto, la opción correcta es la b.</p>
3.	b	<p>De acuerdo a la sección 5.3, para indicar que una clase hereda o se deriva de una clase se usa la forma: Clase Secundaria hereda de Principal.</p> <p>Por lo tanto, la opción correcta es la b.</p>
4.	c	<p>La visibilidad de los atributos de la superclase debe ser protegidos, lo que permite la manipulación de los mismos por parte de la subclase.</p> <p>Por lo tanto, la opción correcta es la c.</p>
5.	b	<p>Si la lógica requiere que la subclase Secundaria tenga sus propios atributos o métodos se los puede declarar e implementar sin problema.</p> <p>Por lo tanto, la opción correcta es la b.</p>
6.	a	<p>La opción A tiene un error de implementación. El método establecerNombresEstudiante de la clase Estudiante recibe como parámetro un valor de tipo Cadena y en la línea e.establecerNombresEstudiante(valor) de la clase Ejecutar, se utiliza la variable valor que ha sido declarada previamente como entera; generando el error.</p> <p>Por lo tanto, la opción correcta es la a.</p>
7.	c	<ul style="list-style-type: none"> ▪ En la opción A se hace uso de un método que no pertenece a la clase EstudiantePresencial (e.establecerNumeroAsignaturas) ▪ En la opción B se utiliza un método que no pertenece a la clase EstudianteDistancia (e.establecerNumeroCreditos) ▪ En la opción C se utiliza un método que si pertenece a la clase EstudianteDistancia (a través del concepto de herencia-e.establecerNumeroCreditos). <p>Por lo tanto, la opción correcta es la c.</p>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 5		
Pregunta	Respuesta	Retroalimentación
8.	c	<ul style="list-style-type: none">▪ En el diagrama de la opción A se está referenciando a dos métodos fuera del alcance en la clase Principal (establecerAtributo3 y obtenerAtributo3).▪ En el diagrama de la opción B, no se usa la relación de herencia conforme la miniespecificación.▪ En el diagrama de la opción C, se cumple con lo indicado en la miniespecificación. Por lo tanto, la opción correcta es la c.
9.	b	<ul style="list-style-type: none">▪ En la opción A se utiliza los métodos agregar y obtener de la clase Secundaria, sin crear o inicializar un objeto de tipo Secundaria.▪ En la opción B se hace uso de los métodos agregar y obtener de la clase Secundaria y además se crear e inicializar un objeto de tipo Secundaria.▪ En la opción C se utiliza los métodos obtener de la clase Secundaria (no se usa los métodos agregar) y además se crear e inicializa un objeto de tipo Secundaria. Por lo tanto, la opción correcta es la b.
10.	b	<ul style="list-style-type: none">▪ La opción que no genera error, es la opción B, por lo tanto, es la correcta.▪ La opción A, está usando los métodos establecerAtributo4 y obtenerAtributo4 que no pertenecen a la clase Secundaria.▪ La opción C, está usando los métodos establecerAtributo3 y obtenerAtributo3 que no pertenecen a la clase Secundaria2.

Ir a la
autoevaluación



Autoevaluación 6		
Pregunta	Respuesta	Retroalimentación
1.	a	En relación a lo indicado en la sección 6.2, la representación de una clase abstracta en diagrama de clases tiene como característica anteponer <<abstracta>> en el nombre de la clase. Por lo tanto, la opción correcta es la a.
2.	b	En relación a la sección 6.1.1 de la guía, se indica que las superclases abstractas son entidades (clases) que tienen al menos un método abstracto que permiten implementar el polimorfismo en las subclases. Por lo tanto, la respuesta correcta es la b.
3.	a	De acuerdo con lo revisado en el apartado 6.2, la forma de representar un método abstracto en un diagrama de clases es: +<<*>> metodo_abstracto(). Por lo tanto, la opción correcta es la a.
4.	a	Una clase abstracta se caracteriza por tener: <ul style="list-style-type: none"> ▪ En su encabezado la palabra reservada abstracta (Clase abstracta Estudiante). ▪ Poseer un método abstracto sin implementación (Método abstracto calcularMatricula() // método sin implementar). La opción que cumple con esos requisitos es la opción a.
5.	c	En una superclase abstracta, se usa el símbolo # (en cada atributo) que indica una visibilidad protegida (protected) para permitir el uso y manipulación por parte de las subclases. Por lo tanto, la opción correcta es la c.
6.	c	La opción correcta es la c, debido a que hace uso de los requisitos necesarios para representar una relación entre una superclase abstracta y una subclase. <ul style="list-style-type: none"> ▪ La clase abstracta Estudiante, está bien estructurada. ▪ La clase Alumno, en su encabezado utiliza la palabra reservada "hereda", para indicar que es subclase de la superclase Estudiante. ▪ La clase Alumno, implementa el método abstracto de su superclase abstracta Estudiante (Método calcularMatricula()), importante mencionar que no se debe incluir la palabra abstracto en la implementación del método en la subclase.
7.	b	Con base en el punto 6.1. de la guía, que se refiere al concepto de polimorfismo; la opción correcta es la b.
8.	c	En la sección 6.1.1 se manifiesta que las clases abstractas no se pueden instanciar, solo permiten derivar subclases. Por lo tanto, la opción de respuesta correcta es la c.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Autoevaluación 6		
Pregunta	Respuesta	Retroalimentación
9.	a	<p>La representación de la relación entre las subclases y superclase abstracta se realiza con una línea que en uno de sus lados termina en flecha. La flecha parte de la subclase y apunta a la superclase abstracta.</p> <p>Por lo tanto, la respuesta correcta es la a.</p>
10.	a	<p>La opción correcta es la a, debido a que utiliza los requisitos necesarios para representar la clase Estudiante1 como una subclase de la superclase Estudiante.</p> <ul style="list-style-type: none">▪ La clase Estudiante1, en su encabezado hace uso de la palabra reservada "hereda", para indicar que es subclase de la superclase Estudiante.▪ La clase Estudiante1, implementa el método abstracto de su superclase abstracta Estudiante (Método promedio()), importante mencionar que no se debe incluir la palabra abstracto en la implementación del método en la subclase.

[Ir a la autoevaluación](#)



5. Referencias bibliográficas

Cadenhead, R. (2014). Java 8. Madrid; España: Anaya Multimedia.

Darwin, I. (2014). Java Cookbook. Sebastopol, USA: O'Reilly.

Deitel, H. M., y Deitel, P. J. (2012). Cómo programar en Java. Ciudad de México, México: Pearson Educación.

Elizalde, R. (2018). Guía didáctica de Programación Orientada a Objetos. Loja, Ecuador: Universidad Técnica Particular de Loja.

Fernández, A. (2013). Python 3 al descubierto. Ciudad de México, México: Alfaomega.

Guardati, S. (2016). Estructuras de datos básicas: programación orientada a objetos con Java. Ciudad de México, México D.F. / Barcelona, España: Alfaomega / Marcombo.

López, L. (2013). Metodología de la programación orientada a objetos. Ciudad de México, México: Alfaomega.

Phillips, D. (2015). Python 3 Object-oriented Programming. Birmingham, UK: Packt Publishing.