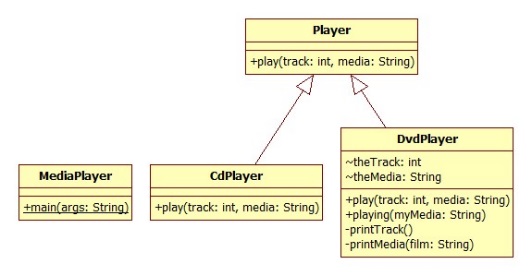
|  |
| --- |
| Entregable #1: Practicando Programación OO |

|  |  |
| --- | --- |
| **Nombre**:  <anotar nombre del estudiante> | **Matrícula**:  <anotar matrícula del estudiante> |
| **Nombre del curso:**  Programación Orientada a Objetos | **Nombre del profesor**:  Luis Ernesto Rubio Torres |
| **Fecha**: |  |
| **Bibliografía**:  Sun Educational Services. SL275. Java Programming Language. USA: Sun  Sun Educational Services. Object Oriented Analysis & Design using UML. USA: Sun | |

*Después de haber trabajado los contenidos de la carpeta:*

*CONTENIDOS SEMANALES -> SEMANA 2*

1. ***Para aprender/practicar dar órdenes a objetos****: Descarga a tu computadora las instrucciones de acceso al “Laberinto Clásico.pdf” del sitio Code.org, y resuelve los 20 rompecabezas ahí indicados.*
2. ***Para aprender/practicar la lectura e identificación de elementos de código OO Polimorfo****: Descarga y corre en tu computadora, el programa anexo: MediaPlayer.java*
   1. *Observa que su Jerarquía de Herencia, corresponde con el siguiente diagrama de clases en UML:*

**

* 1. *Descarga a tu computadora el documento PlantillaDeExploracion.doc anexo, y completa los espacios marcados con X, después de estudiar y hacer funcionar el programa MediaPlayer.java*

1. ***Para aprender/practicar la escritura de código OO Polimorfo****: Descarga, completa lo requerido y corre en tu computadora, el programa anexo: Mascotas.java reusando la “lógica” de MediaPlayer.java*
2. ***Para aprender/practicar el proceso elemental de programación OO****: Descarga a tu computadora la especificación de requerimientos “La Revisión de Salarios.pdf”, y aplicando el método EEPOO analiza, diseña y codifica en Java el programa requerido.*

***Nota****: puedes auxiliarte del documento anexo: “Guía Solución EEPOO Revisión Salarial.pdf”*

***Nota****: para efectos de aprendizaje, usa la guía, sólo después de haber intentado por ti mism@ cada ejercicio.*

*Con base a tus resultados previos, responde a las siguientes preguntas:*

1. ***Captura imágenes con tus resultados al completar los Formatos A: la Jerarquía de Clases y B: Polimorfismo, de tu documento PlantillaDeExploracion y pégalas en el siguiente espacio (valor 2 %).***

|  |
| --- |
|  |

***2) Captura imágenes con tus resultados al completar la funcionalidad del programa Mascotas.java, explicando a detalle el código de tu programa y cómo aplica los principios de Abstracción, Herencia y Polimorfismo (valor 2 %).***

|  |
| --- |
| ***Codigo:***  ***//Mascotas.java***  ***public class Mascotas{***  ***public static void main(String[]args){***  ***System.out.println("Mis Mascotas");***    ***//¿Qué línea hace falta aquí para que funcione este programa?***  ***Mascota miMascota;***    ***miMascota = new Gato();***  ***miMascota.come("ratones"); //despliega: El gato come ratones***  ***miMascota.come("whiskas"); //despliega: El gato come whiskas***    ***miMascota = new Perro();***  ***miMascota.come("huesos"); //despliega: El perro come huesos***  ***miMascota.come("carne"); //despliega: El perro come carne***    ***miMascota = new Conejo();***  ***miMascota.come("alfalfa"); //despliega: El conejo come alfalfa***  ***miMascota.come("zanahorias"); //despliega: El conejo come zanahorias***  ***}***  ***}***  ***/¿Qué código hace falta agregar para que funcione este programa?***  ***class Mascota{***  ***public void come(String alimento){***  ***System.out.println("La mascota come " + alimento);***  ***}***  ***}***  ***class Gato extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El gato come " + alimento);***  ***}***  ***}***  ***class Perro extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El perro come " + alimento);***  ***}***  ***}***  ***class Conejo extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El conejo come " + alimento);***  ***}***  ***}***  ***------------------------------------------------------------------------------------------------------------------------------------------***  ***Resultado de ejecución:***  ***Mis Mascotas***  ***El gato come ratones***  ***El gato come whiskas***  ***El perro come huesos***  ***El perro come carne***  ***El conejo come alfalfa***  ***El conejo come zanahorias***  ***------------------------------------------------------------------------------------------------------------------------------------------***  ***Explicación:***  ***El programa se inicia en el método main de la clase Mascotas, la cual crea una variable de tipo Mascota, a a la cual se le asigna el subclases de la clase Mascota que son Gato, Perro y Conejo, las cuales sobrecargan la función come()m, que lo que hace es desplegar un mensaje en la consola que indica que la mascota en particular como alimento que le es pasado como parámetro.***  ***Principio de abstracción:***  ***El principio de abstracción se puede ver en la practica con las siguientes líneas:***  ***1 Mascota miMascota;***  ***2***  ***3 miMascota = new Gato();***  ***4 miMascota.come("ratones"); //despliega: El gato come ratones***  ***5 miMascota.come("whiskas"); //despliega: El gato come whiskas***  ***6***  ***7 miMascota = new Perro();***  ***8 miMascota.come("huesos"); //despliega: El perro come huesos***  ***9 miMascota.come("carne"); //despliega: El perro come carne***  ***10***  ***11 miMascota = new Conejo();***  ***12 miMascota.come("alfalfa"); //despliega: El conejo come alfalfa***  ***13 miMascota.come("zanahorias"); //despliega: El conejo come zanahorias***  ***Como puede verse en la línea 1 se crea un objeto de tipo Mascota, sin embargo se le asignas objetos de tipo Gato , Perro y Conejo en las líneas 3, 7 y 11 respectivamente, y al ejecutarse el método come() vemos que lo que en realidad se ejecuta es el método correspondiente a las subclases(Gato, Perro, y conejo), no de la clase padre(Mascota.).***  ***Herencia:***  ***Aquí la herencia se aplica al definir las clases Gato, Perro y Conejo que extienden la clase Mascota, heredando sus métodos y propiedades , como podemos ver en las siguientes líneas:***  ***class Gato extends Mascota{***  ***class Perro extends Mascota{***  ***class Conejo extends Mascota{***  ***Polimorfismo:***  ***El principio de polimorfismo podemos verlo aplicado al definir un nuevo método come(String alimento) en las subclases Gato, Perro y Conejo que sustituyen al método originario de la clase Mascoat, como se puede ver en las siguientes lineas:***  ***class Gato extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El gato come " + alimento);***  ***}***  ***}***  ***class Perro extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El perro come " + alimento);***  ***}***  ***}***  ***class Conejo extends Mascota{***  ***public void come(String alimento){***  ***System.out.println("El conejo come " + alimento);***  ***}***  ***}*** |
|  |

***3) Inserta las imágenes de tus resultados al desarrollar la etapa de análisis para el sistema de “La Revisión de salarios” (incluyendo los diagramas UML de casos de uso, de instancias y de clases a nivel de análisis), explicando tu experiencia al desarrollar estos modelos (valor 4 %).***

|  |
| --- |
| **User View**  1.-Lo que puede hacer el usuario con el sistema  2.-Se utilizaran 2 clases:  1.- RevisionSalarial.java.- Que contiene el fujo de actividades a realizar para desplegar la revisar salarial de los empleados solicitados, y lanza el programa mediante el método main()  2.- Empleado.-Que contiene los datos necesarios para calcular la revisión salarial de los empleados.    Lo importante de esta fase es generar una visión de inicio de las actividades que puede ejecutar el usuario dentro del sistema y la estructura de las clases asi como sus responsabilidades necesarias para instrumentar el sistema. |

***4) Inserta las imágenes de tus resultados al desarrollar la etapa de diseño para el sistema de “La Revisión de salarios” (incluyendo los diagramas UML de secuencias y/o colaboración y de clases a nivel de diseño), explicando tu experiencia al desarrollar estos modelos (valor 4 %).***

|  |
| --- |
| **1.-Para realizar las actividades necesarias para presentar el cuadro de revisión salarial de los empleados se generaron los siguientes diagramas:**  2.-para crear las clases correspondientes a los empleados el diagrama de secuencias:  3.-Para presentar la revisión salarial:  \*\*Unicamente se presenta el diagrama para el empleado 3, se omiten los de empleado1 y empleado2, que serian exactamente igual. |

***5) Incluye las imágenes de tus resultados al desarrollar la etapa de codificación para el sistema de “La Revisión de salarios” (código de la aplicación completa en Java funcionando en un IDE), explicando tu experiencia al programar este software (valor 4 %).***

|  |
| --- |
| **Clase RevisionSalarial.java**:    **Empleado.java:**      El desarrollo del sistema fue sencillo a partir del análisis previo, junto con la guía de solución que guuiaron la codificación en NetBeans  **Ejecución del programa:** |

***6) Completa los 20 ejercicios (rompecabezas) del “Laberinto Clásico” del sitio Code.org y explica brevemente como los resolviste, comentando tus resultados y los videos del sitio. Incluye la imagen del Certificado de Terminación personalizado con tu nombre, que obtendrás al completar los ejercicios (valor 4 %).***

|  |
| --- |
|  |

|  |
| --- |
| ***Tip:***  ***Instalando herramientas para modelar aplicaciones con UML:***  *Si lo deseas los modelos UML pueden hacerse y presentarse “a mano alzada”, aunque para facilitarte el proceso de modelado de soluciones OO, puedes probar la instalación de una o varias de las herramientas de diagramación siguientes - o utilizar alguna otra de tu preferencia. Su empleo muy intuitivo.*  *Accede a sus páginas de instalación mediante los enlaces:*  *instalar: StarUML:* [***http://staruml.io/***](http://staruml.io/)  *instalar: StarUML v.5.0:* [***http://staruml.sourceforge.net/v1/download.php***](http://staruml.sourceforge.net/v1/download.php)  *instalar: ArgoUML:* [***http://argouml.tigris.org/***](http://argouml.tigris.org/)  *instalar: FrameUML v.1.2.1:* [***https://sourceforge.net/projects/frameuml/***](https://sourceforge.net/projects/frameuml/) |