**HOJA INTERACTIVA DE APRENDIZAJE Nº 05**

**Apellidos y Nombres:** Casas Moreno Christian Joel

**Docente:** Nestor Audante Ramos **Fecha de entrega:** 30/10/2021

**INTRODUCCIÓN**

1. **Indicadores:**
   1. Identifica el uso de interfaces
   2. Identifica el comportamiento de las enumeraciones.
   3. Crea clases que implementan interfaces.
   4. Crea clases con métodos que usan enumeraciones y excepciones.
2. **Actividades para desarrollar en forma individual:** 
   1. Complete la tabla sobre la sintaxis Java:

|  |  |
| --- | --- |
| **Código fuente** | **Errores** |
| public enum DIAS\_SEMANA{  JUEVES(5), LUNES(0);  int dia;  } | * El nombre de la clase no puede llevar un espacio. * Las constantes no pueden tener parámetros de entrada. * Tampoco se le pueden asignar valores. |
| public interface T2 {  abstract void imprimir(){  }  } | * El método no puede ser implementado en una interface. |
| package com.tesla.\*;  public class MiExcepcion extends Object{  Exception(String mensaje){  Super(mensaje);  }  } | * Object no tiene padre, por lo que no se puede llamar a un super(). * Exception no puede recibir parámetros de entrada. |

* 1. Complete la tabla escribiendo los conceptos aprendidos en clase

|  |
| --- |
| **Conceptos** |
| **Clase abstracta**  Es aquella clase que no se puede instanciar. No se pueden crear objetos a partir de esta clase. La finalidad de una clase abstracta es servir como clase base para otras clases a las que generalmente se conoce como clases "concretas". |
| **Interface**  Es una colección de métodos abstractos y propiedades constantes. En las interfaces se especifica qué es lo que se debe hacer, pero no su implementación. Serán las clases que implementen estas interfaces las que describen la lógica del comportamiento de los métodos. |
| **Consideraciones de una clase al implementar una interface**   * A diferencia de la herencia, se pueden implementar más de una interface a la vez. * Una interface se puede heredar de otra interface. * Se pueden poner atributos, los cuales actuarán como constantes en las clases que los implementen. |
| **Excepción**  Son los errores cuando se está ejecutando un programa. Estos ocurren cuando se produce un error en alguna de las instrucciones de nuestro programa. Cuando se produce una excepción se muestra en la pantalla un mensaje de error y finaliza la ejecución del programa. |
| **Consideraciones para construir las secciones catch**   * printStackTrace te muestra el error que existe en try. * El Exception debe ir al final, pues es el más genérico y ponerlo al inicio invalidaría al resto de excepciones haciendo que sean código muerto. * Al finalizar las instrucciones del/de los catch, se puede pasar al finally, siendo que lo que se encuentra en finally siempre se ejecuta independientemente de lo que hay en el try o en el catch. |
| **Enumeración**  Es una lista de constantes con nombre que definen un nuevo tipo de datos. Un objeto de un tipo de enumeración solo puede contener los valores definidos por la lista. Por lo tanto, una enumeración le brinda una manera de definir con precisión un nuevo tipo de datos que tiene un número fijo de valores válidos. |

* 1. Complete la tabla.

|  |
| --- |
| **Código fuente: Cree una clase con un método que maneje una excepción y otra que lance la misma excepción.** |
| public class ExcepcionEjemplo extends RuntimeException{  public ExcepcionEjemplo() {  super();  }  public ExcepcionEjemplo(String message) {  super(message);  }  }  class ClaseLanzarExcepcion{  public void mayoriaEdad(int edad){  if (edad<18){  throw new ExcepcionEjemplo("La persona no puede ser menor de edad.");  }  }  public static void main(String[] args) {  int edadJuanito = 12;  ClaseLanzarExcepcion c1 = new ClaseLanzarExcepcion();  c1.mayoriaEdad(edadJuanito);  }  } |

* 1. Caso de estudio: Interfaces

**Parte I**

Un estudiante de primaria le pide que escriba un programa para calcular el área y el perímetro de polígonos regulares como muestra la tabla:

|  |  |  |
| --- | --- | --- |
| Polígono regular | Área | Perímetro |
| Triángulo |  |  |
| Cuadrado |  |  |
| Pentágono |  |  |
| Hexágono |  |  |
| Heptágono |  |  |

**Defina las clases necesarias, donde una clase abstracta para figura es requerida para resolver el pedido del estudiante.**

**Parte II**

**Reutilice las clases anteriores y reemplace el uso de la clase abstracta con una interface para resolver el pedido del estudiante.**

* 1. Suba el documento resuelto y la solución del caso de estudio en un directorio llamado lab-5.

1. **Referencia de material** 
   1. Geeksforgeeks (2021). Interfaces in Java. Recuperado de <https://www.geeksforgeeks.org/interfaces-in-java/>
   2. Wlaton (2020). Manejo de Excepciones en Java. Recuperado de <https://javadesdecero.es/intermedio/manejo-de-excepciones/>

1. **Conclusiones y recomendaciones**