ESTRUCTURA DE DATOS

Hito 2

1. ¿A que se refiere cuando se habla de POO?

El modelo orientado a objetos (OO) es una colección de objetos o clases que permite que un programa pueda examinar y manipular elemento de su entorno. Es modelar un problema existente con entidades independientes que interactúan entre sí.

2. ¿Cuáles son los 4 componentes que componen POO?

Son: Abstracción, Encapsulamiento, Modularidad y Jerarquía.

3. ¿Cuáles son los pilares de POO?

Son: Herencia, Polimorfismo, Abstracción y Encapsulamiento.

4. ¿Qué es Encapsulamiento y muestre un ejemplo?

Decimos que el encapsulamiento en la programación orientada a objetos es cuando limitamos el acceso o damos un acceso restringido de una propiedad a los elementos que necesita un miembro y no a ninguno más.
El elemento más común de encapsulamiento son las clases, donde encapsulamos y englobamos tanto métodos como propiedades.
Otro ejemplo muy común de encapsulamiento son los getters y setters de las propiedades dentro de una clase. Por defecto nos dan el valor “normal” pero podemos modificarlos para que cambie.

5. ¿Qué es Abstracción y muestre un ejemplo?

La abstracción es un concepto muy similar al de la encapsulación, con la diferencia principal de que la abstracción nos permite representar el mundo real de una forma mas sencilla. Podríamos definirlo también como la forma de identificar funcionalidades necesarias sin entrar en detalle de lo que estamos haciendo.

El ejemplo mas claro es cuando frenamos en el coche, para nosotros, el usuario, es únicamente pisar el freno, pero por detrás el coche realiza una gran cantidad de acciones. También imaginad que en todos los coches el conductor frena de la misma manera, independientemente de si los frenos son de disco o de tambor. Mientras que el coche realiza diferentes acciones.

6. ¿Que es Herencia y muestre un ejemplo?

La Herencia es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la ***Abstracción***, ***Encapsulación*** y ***Polimorfismo***. Al principio cuesta un poco entender estos conceptos característicos del paradigma de la POO porque solemos venir de otro paradigma de programación como el paradigma de la programación estructurada (ver la entrada"Paradigmas de Programación), pero se ha de decir que la complejidad está en entender este nuevo paradigma y no en otra cosa

El ejemplo que proponemos es un caso en el que vamos a simular el comportamiento que tendrían los diferentes integrantes de la selección española de futbol; tanto los Futbolistas como el cuerpo técnico (Entrenadores, Masajistas, etc…). Para simular este comportamiento vamos a definir tres clases que van a representaran a objetos Futbolista, Entrenador y Masajista. De cada unos de ellos vamos a necesitar algunos datos que reflejaremos en los ***atributos*** y una serie de acciones que reflejaremos en sus ***métodos***. Estos atributos y métodos los mostramos en el siguiente diagrama de clases:

7. ¿Qué es Polimorfismo y muestre un ejemplo?

En **programación orientada a objetos**, polimorfismo es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación.

Un ejemplo clásico de **poliformismo** es el siguiente. Podemos crear dos clases distintas: Gato y Perro, que heredan de la superclase Animal. La clase Animal tiene el método abstracto makesound() que se implementa de forma distinta en cada una de las subclases (gatos y perros suenan de forma distinta).

8. Que es un ARRAY?

Un array Java es una estructura de datos que nos permite almacenar una ristra de datos de un mismo tipo. El tamaño de los arrays se declara en un primer momento y no puede cambiar en tiempo de ejecución como puede producirse en otros lenguajes.

9. ¿Qué son los paquetes en JAVA?

Un **Paquete** en Java es un contenedor de clases que permite agrupar las distintas partes de un programa y que por lo general tiene una funcionalidad y elementos comunes, definiendo la ubicación de dichas clases en un directorio de estructura jerárquica.

10.¿Cómo se define una clase main en JAVA y muestra un ejemplo?

El método Main es el punto de entrada de una aplicación de C# (las bibliotecas y los servicios no requieren un método Main como punto de entrada). Cuando se inicia la aplicación, el método Main es el primero que se invoca

```
// Clase principal iniciadora del programa ejemplo aprenderaprogramar.com
public class TestDeposito {
   public static void main (String [ ] args) {

      //Aquí las instrucciones de inicio y control del programa

      System.out.println ("Empezamos la ejecución del programa");

   } //Cierre del main
} //Cierre de la clase
```

11. Generar la clase Provincia.

```
public class Provincia {

   public String nombre;

   public Provincia (){
      nombre = "Caranavi";
   }

   public String getNombre(){
      return this.nombre;
   }

   public void setNombre(){
      this.nombre = nombre;
   }

   public void muestraProvincia(){
      System.out.println("Provincia: " + this.nombre);
   }
}
```

12.Generar la clase Departamento.

```
public class Departamento {
   public String nombre;
   public nroDeProvincias[] Provincia;

   public Departamento(){
      nombre = "La Paz";
   }
   public String getNombre(){
      return this.nombre;
   }

   public nroDeProvincias[] getProvincia(){
      return this.Provincia;
   }

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public void setProvincia(nroDeProvincias[] provincia) {
      Provincia = provincia;
   }
   public void muestraDepartamento(){
      System.out.println("Nombre: "+ this.nombre);
   }
   private class nroDeProvincias {
   }
}
```

13.Generar la clase País.

```
public class Pais {

   public String nombre;
   public int nroDepartamentos;
   public departamentos[] Departamento;


   public Pais() {

      nombre = "Bolivia";
      nroDepartamentos = 9;

   }


   public Pais(String nombre,int NroDeppartamentos, departamentos[] Departamento) {
      this.nombre = nombre;
      this.nroDepartamentos = nroDepartamentos;
      this.Departamento = Departamento;
   }


//getters
public String getNombre() {
   return nombre;
}

public int getNroDepartamentos() {
   return nroDepartamentos;
}

public departamentos[] getDepartamento() {
   return Departamento;
}
```

```
   //setters

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public void setNroDepartamentos(int nroDepartamentos) {
      this.nroDepartamentos = nroDepartamentos;
   }

   public void setDepartamento(departamentos[] departamento) {
      Departamento = departamento;
   }

   public void muestraPais(){
      System.out.println("Nombre: "+ this.nombre);
      System.out.println("Numero de Departamentos: "+ this.nroDepartamentos);
   }
   private class departamentos {
   }
}
```

No pude completarlo ☹