JAVA POO  
 ASOCIADOS  
 UML  
  
  
  
  
  
  
  
Que es un objeto?  
  
En pocas y simple palabras le podria decir que un objeto es la instancia de una clase pero esta descripcion es muy pobre para lo que tenemos delante ya que estan son las bases de poo, un objeto es donde inicializamos los atributos y verbos de nuestra clase algo que debemos recordar es que desde ahora escucharemos mucho esta palabra ya que por medio de esta logramos modular, reutilizar y llevar un orden nuestro codigo separando por objetos.  
Debemos tener en cuenta para que algo sea un objeto debe tener propiedades y acciones contamos con dos tipos de esto que son:  
  
Objetos conceptuales como nos dice su palabra concepto son aquella que no podemos por si ver como funcionan pero estan hay o hasta incluso cosas que viven en nuestra por ejemplo imaginemonos como llega la luz al foco o como funciona una placa electronica o un motor.  
  
Objetos Fisicos estos son todos aquellos que podemos ver y tocar que estan a nuestro alrededor puede ser desde un silla, nuestro, escritorio o nuestra casa.  
  
  
Muchos se suelen confundir saben los atributos tambien son propiedades estas en pocas palabras son caracteristicas de nuestro obejto por ejemplo tenemos un cuaderno o diccionario una de sus propiedades seria color, numero de paginas y el topic o tema de este en lo que respeta a sus acciones podemos decirle que responda aca que le abren o lo buscan sin mas sigamos con el siguiente tema pero antes veamos un ejemplo a codigo.  
  
  
esta es nuestra superclase o conocida padre con sus respetivos atributos y acciones

*class* Printing{

*String* id;

*String* Topic;

*String* NumberPage;

*public* *void* PrintingTalk(){

System.out.println("eh estoy vivo");

}

*public* Printing(*String* *id*, *String* *topic*){

*this*.id = id;

*this*.Topic = topic;

}

}

Pero no nos olvidemos de su clase main donde es la entrada de todo como querida esta debemos tener clase no hay una sola forma de iniciar un constructor este no devuelve valores pero puede iniciar sin nada esto lo veremos mas adelante por el momento veremos como queda la clase principal.  
  
  
  
  
  
  
  
  
  
 Que es una abstracion?  
  
Lo podria decir en pocas palabras creamen siempre pofundizar sera lo mejor la misma palabra los dice abstraer pero que extraemos. Los atributos y acciones que nos sirven para modular, simplicar y comprender que es lo que estamos haciendo o queremos hacer con el obejto que tenemos presente como es cuales son sus acciones y de que tiene es el objeto esto nos ayuda mucho no se si ustedes recuerda la frase “divide y venceras ” en se enfoca la poo y sus bases fundamentales y estamos frente una de ellas.  
  
Pero porque estan importante como ya sabemos este de tipo de programacion se centra en la reutilizacion de codigo ya que en algun momento de la historia de se llego a creer que mientras mas lineas de codigo mejor el programa pero no por tal al momento de modular y hacer una instancia de cierto obejto se usa la abstracion para poder contar con cosas fundamentales de este.  
  
Para que ustedes como yo puedan comprender mejor esto seguimos con el ejemplo del diccionario al usar este principio abstraemos atributos y acciones de nuestro objeto veamos el ejemplo:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 MODULARIDAD  
  
Estoy mas que seguro que para este punto usted como desarrollador ya sabe lo que es la modularidad y en caso que no seas asi estamos siempre dispuestoa darte una mano .  
Si usted ya maneja este concepto por favor pase a la siguiente pagina.  
  
Sin mas que decir empecemos la modularidad es algo increible creame sera de mucha ayuda recuerda en la programacion estructurada cunado alguna linea de codigo tenia un erro todo se dañaba pues gracias a este concepto que nos permite separar codigo y que cada uno tenga una funcion si alguna parte de nuestro codigo llegara a fallar no se dentria por completo esto tiene muchos beneficio leamos a continuacion.  
  
Reutilizacion de codigo es uno de los primeros, la escalabilidad del proyecto mantemiento ,es decir, cualquier con los conocimientos previos podra leer nustro codigo para poder entender esto veamolo como un juego de legos donde cada uno se puede agrupar. Y hacer grandes cosas pero si una parte de este se daña claro que senterimos el daño pero sabremos en cuention de minutos donde estos nos permite tener una vista ordenda de nuestros programas y ver sus errores a plena vista es por tal que la modularidad es fundamental en la programacion orientada a objetas es aquí donde veremos reflejado el divide y venceras les dejare un pequeño de un codigo modularizados con su respetiva explicacion.  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Metodo Constructor   
  
  
  
Un constructor es la puerta a muchos lugares este es un metodo que utilizara el nombre no siempre es necesario declarar uno ya que el compilador no da uno por defecto, este nos ayudara a crear nuevas instancias , este tiene el nombre de la clase que se esta incializando pero ojo esto puede variar dependendiendo de mucho factores que veremos mas a futuro este se llamaba con la palabra new. Y con () que nos hacen una llamada que es un metodo, este nunca regresaran un valor pero si podems agregarle dentro de los parentesis pero como se ve un metodo constructor veamos. Un ejemplo:  
  
  
  
  
  
  
  
  
  
  
  
  
este puede tener 0. O mas argumento hasta null podemos tener nuestro metodo constructor este se le constructor   
  
  
  
  
  
  
  
Static variables y metodos static   
  
  
Los metodos static son aquellos que estan disponibles para toda la clase u otras pero como es posible esto lo que hace la palabra static es que le da un nivel de acceso. Que todo le puedan llamar y acceder a este, esta es una palabra reservada que quiere decir esto que ningun metodo se puede llamar asi ni declarar una clase asi pero como llamamos a esto metodos con el nombre de la clase.nombredelmetodostatic no necesitamos instanciar , podemos crear metodos static veamos un ejemplo  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Final variables Constantes  
  
  
Que nos dice la palabra final en java esta palabra nos dice que no podemos modificar datos o varibles o metodos que declares con la palabra final es decir sera una constantes inmutable ,es decir, no podremos modificar esta variable o metodo veamos ejemplos para poder comprender:  
  
  
  
   
  
  
  
  
  
  
  
  
Cabe recalcar que la palabra final es una palabra reservada y que casi siempre le encontraremos con la static pero ojo no es una regla escrita .  
En que caso se usa la palabra final? Por que o para que necesitariamos datos constantes. En nuestro codigo son excelentes preguntas primero en el caso que necesitemos x datos sean inmutables y que no puedan cambiar nunca por que necesitamos esto por que nos es fatible tener datos de esta forma   
  
  
 Sobrecarga de metodos y constructores  
  
  
Que son estos en varias ocasiones necesitamos que nuestros metodos tengan el mismo nombre pero con diferentes metodos argumentos la sobrecarga de metodos nos dicen que indepenediente de los datos enviados estos se inicializara un de estos metodos cuando se le envien los datos con lo que se incializa o sus argumentos veamos ejemplos para poder comprender:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Por otro lado tambien podemos sobrecargar los constructores   
 veamos como podemos hacerlo:

*class* Main{

*public* *static* *void* main(*String*[] *args*){

*Printing* printing = new Printing("11", "english-spanish");

printing.PrintingTalk();

}

}

Veamos al abstraccion de esta forma creamos nuestra clase  
  
  
Clase Dictionary{  
  
//aquí extraemos sus atributos y acciones para poder modular y hacer muchas mas cosas con estos  
  
String id;  
String topic;  
String NumberPage;  
  
  
public void DictionaryAnswer(){  
System.out.println(“hey”)  
  
  
  
}  
  
  
  
  
}

Class Animal{  
  
  
  
 Animal animalOne = new Animal();  
//en otro caso tenemos un metodo constructor normal pero veamos otro ya con datos  
  
  
Animal animalOne = new Animal(name : “ ”, age: “ ”, stature ; );  
}

string static daticOne = “hey”;

Int final datosOne = 1;

Public void int varibleDatos(int a , int b){  
  
  
}  
public void string variableDatos(String a , String b){  
  
  
}  
  
public void float varibleDatos(float a, float b){  
  
  
}

public void double varibleDatos(Double a , Double b){  
  
}

digamos que tenemos nuestro constructor en nuestra main  
  
  
class main{  
public static void main(String[] args){  
  
 edificio myedificio = new edificio();   
  
}  
}  
  
class edificio{  
  
  
  
edificio(){  
System.out.println(“esta es un prueba para tu futuro solos sigue recuerda el obejtivo”);  
}  
  
edificio(){  
   
String dato = “ nueve datos solamente “;  
  
}  
  
  
}

Modificadores de acceso se que para este momento ya hemos hablado de esto pero no esta demas recordarlo nuevamnete y dar una sumercion en estos en base a que nos permiten y a que no  
  
Mofidicadores   
  
  
public este no deja todo libre desde clase, package(paquetes) , subclase(hija o hijo), otros  
  
protected este nos da acceso a clase, package , subclases pero quita el acesso a otros  
  
  
default este nos deja aceso a clase y package no mas.  
  
private este solo nos deja acesso a claagregar el resto de cosas   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Getter y setter  
  
  
 Para este punto ya tenemos hemos usado los modificadores de acceso y nos nacera duda como llamabamos a nuestros datos o cambiamos sus datos. Los getter nos dicen al español obtener con este podemos imprimir nuestro datos por otro lado tenemos los setter que nos ayudaran a sobrescribir datos de estos o poder cambiar sus datos veamos su sintaxi para poder hacer uso de estos metodos reservados.  
Algo que acotar es que estos van de la mano con el encapsulamiento y que son necesarios para llevar un codigo robusto y estable de tal forma que tenga seguridad y validacion.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Variable vs obejtos  
  
  
Que son las variables esta son todos aquellos datos primitivos que nos hemos encontrado hasta hoy que hemos trabajo y estudiado donde se almacenan estos en el stack o pila como se almacenan veamolo de esta forma son como libros uno encima de otro el primero es el ultimo y el ultimo es el primero estos son datos simples pero no por esto son menos.  
Por otro lado tenemos los objetos se nos dicen que multiples cosas para hacer un objeto pero a que nos referimos con “multiples cosas” para poder comprender este punto tenemos que regresar a todo lo aprendido que componen a un objeto un clase que es la que define a este y es de donde extraemos sus atributos y acciones y donde le declaramos, modulamos tambien tenemos todos sus metodos y caracteristicas como la herencia el polimorfismo entre otros, pero se que a mucho se le vendra la pregunta por que necesitamos saber las diferencia de aquellos es muy comun su pregunta desde ahora trabajaremos de la mano de los dos. Para poder crear codigo robusto enfocado en interfazes  
Pero se me escapa algo como almacenamos los objetos imagemos una gran bibliotecas y cajas que se adpatan a la necesidad de los objetos que tenemos es asi como los almacenamos sin tener mayor errores estos son almacenado en el heap o al español el monto hace mucha referencia al lugar donde son almacenados.  
  
  
  
  
  
  
Veamos ejemplos y unas cuantas caracteristicas tenemos datos simples que contiene un valor directo  
  
  
int x = 0;  
  
  
 Por otro lado tenemos los obejtos mas complejos mas grande y estos como ya lo mencione tiene muchas caracteristicas y se almacenan en el heap  
  
example:  
  
  
  
Seres oneSeres = new Seres();

Class Ejemplo{  
private String ejemploOne;  
private String ejemploO;  
private Double datosOne;  
   
  
public ejemplo(String ejemploOne, String ejemploO ){  
 this. ejemploO = ejemploO;  
 this. ejemploOne = ejemploOne;  
  
}  
  
  
  
  
  
public void GetejemploOne(String ejemploOne){  
 return ejemploOne;  
}  
 public void SettejemploO(String ejemploO){  
 return ejemploO;  
}  
  
  
}

Cuando usar cada una de estas es facil y sencilo cuando tengamos datos simples, cunado no necesitemos modular cosas complejos y calculos basicos para variables.  
  
  
En el caso de obejetos para cosas del mundo real, relaciones entre entidades .

Clases anidadas  
  
Las clases anidadas se que a muchos se le haran concocido esta palabra y lo se es por los for anidados y si miramos atrás hemos recorrido un inmeso camino y si todavia nos falta demasido.  
 Por otro lado que son este tipo de clases como recordamos en el for se usaban dos para recorrer datos en mas de una dimension y poder imprimir en este caso hacemos uso de este tipo de clases para poder compartir datos crear metodos con mayor seguridad. Flexibilidad entre ellos pero debemos saber que los metodos que usemos dentro de estas clases dependienedo de los modificadres de acceso que usemos tendremos acceso o no y se que les surgira la duda pero no que JAVA solo permite una sola clase en cada archivo pero no es asi al crear este tipo java la vera como una misma clase debido a que esta vive dentro de la otra y tiene un nivel de encapsulamiento mayor que el resto   
   
  
Por otro lado contamos con las clases internas como ya mencione antes estas son aquellas que tiene un mayor nivel de encapsulamiento y tiene acceso tanto a los metodo, atributos y verbos de la clase inicial , se nos dice que las clases locales son aquellas que estan declaradas de un metodo es decir que 100% limitadas por este tanto su nivel de modificadores y mas.  
Se utiliza normalmente cuando se necesita una clase auxiliar.  
  
  
Cuando debemos usar cada una de estas veamos en el caso de las clases internas cuando necesitemos ocultar la implemetacion de la clase recuerdan que dije que estas tiene un nivel de encapsulamiento mayor, una clase auxiliar relacionada con la clase y cuando se necesite acceder a los miembros privados de la clase externa.  
  
  
En lo que respeta a las clases locales cuando necesitemos una clase auxiliar dentro de un metodo y cuando necesites encapsular un comportamiento especifico dentro de un metodo.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Enumerations  
  
  
En pocas palabras estos son los unicos que nos sirven para declarar una collections de constantes por tal debemos escribirles en mayuscula es asi como podremos declarar constante con la palabra reservada enum estas cuentas o pueden tener atributos, metodos y constructores.  
  
  
example  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Herencia Don’t repeat Yourselft   
  
  
  
  
 Que es la herencia imagina que puedas compartir atributos metodos, acciones con una palabra y crear jerarquias de muchas formas pues para eso y mucho mas tenemos la herencia que hace mucha alusión a su nombre ya que heredamos datos a nuestras subclases con una simple palabra reservada. “Extends” pero debemos tener en cuenta que nos heredar por heredar debemos llevar ciertos liniamientos para poder hacer uso de este concepto y que nuestro codigo y herencia sea efectivo tenemos la frase “es un” nuestra clase es ser humano “es un ” ser nuestra frase es coherente por lo tanto nuestra herencia seria efectiva.  
  
Si ponemos atencion su filosofia nos dice que repitamos codigo es debido que al momento que heredamos reutilizamos este que es una de la base de poo eh aquí le he mencionado uno de pilares fundamentales de lo que estamos a punto de conocer pero para ahora es seguro que. Tenemos muchas preguntas que respuestas por tal me adelanto ya sabemos que son los constructores y les eh mostrado ejemplos donde tendremos que hacer referencia a los datos de nuestra clase padre o simplemente incializarlo en esta para estas tareas tenemos las palabras reservadas “THIS” y “SUPER”.   
  
  
This nos ayudara incializar nuestros atributos desde la super clase es decir con esta incializamos nuestro constructor.  
  
  
Super es quien nos ayudara a hacer referencia a nuestra clase padre cuando tomamos sus datos desde una subclase o clase hija es decir diria “padre ” este es tu atributo de parte de la herencia.

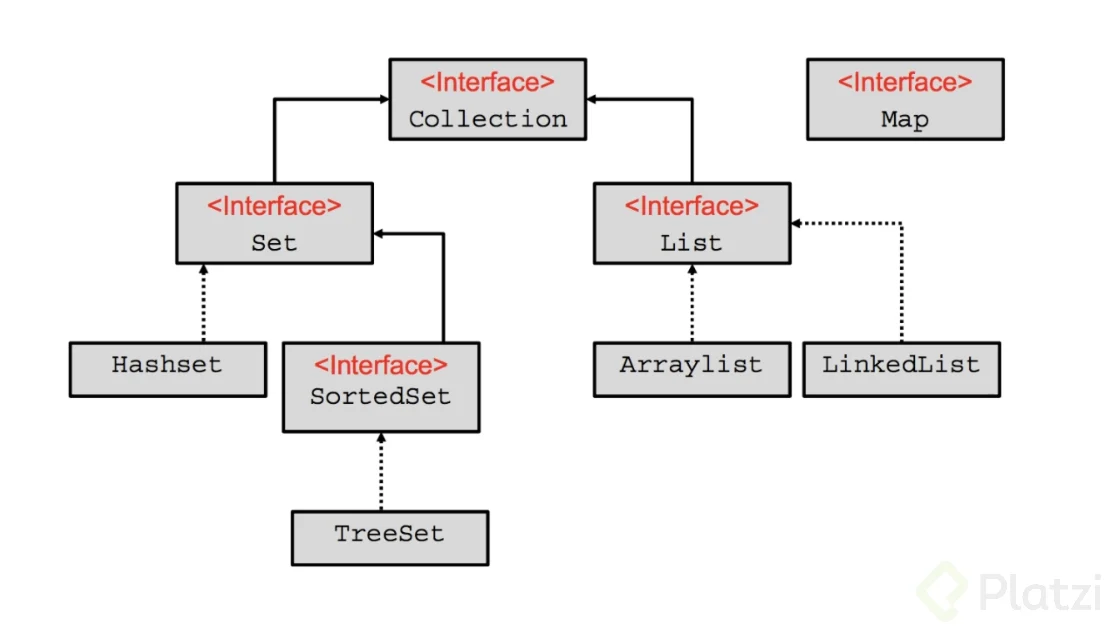
Public enum MONTH{  
  
 ENERO, FEBRERO, MARZO, ABRIL , MAYO , JUNIO , JULIO, AGOSTO, SEPTIMEBRE, NOVIEMBRE   
  
  
}

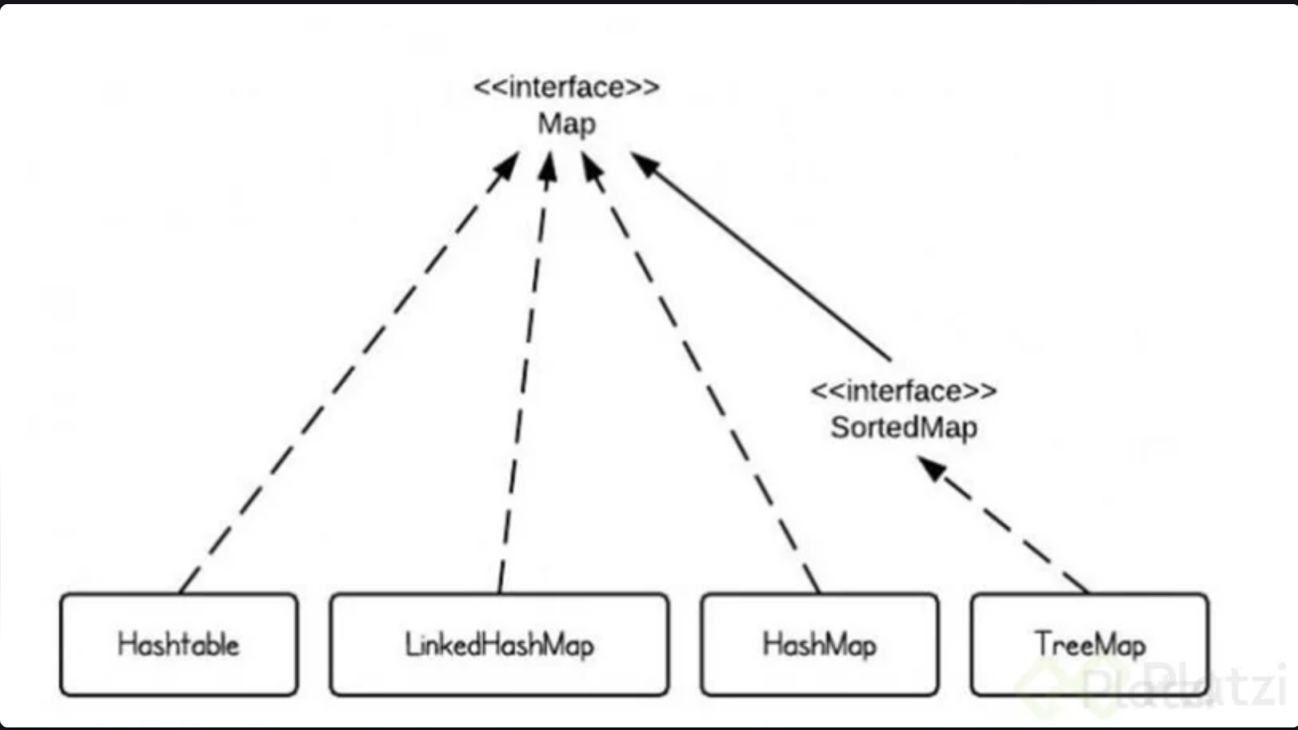
Class transporte{  
Integer ruedas;  
Double peso;

String name;  
  
  
public transporte(Integer ruedas, Double peso, String name){  
 this.ruedas = ruedas;  
 this.peso = peso;

This.name = name;  
  
}  
  
}  
  
  
  
Class auto extends transporte{  
  
  
public auto(Integer rueda, String name){  
 Super(rueda, name);  
  
}  
}

Polimorfismo  
  
  
  
  
Este es otro pilar fundamental nos da un metodo para no repetir codigo sin de manera simple y rapida pero debemos. Cumplir con ciertas cosas que nuestros metodo no sea static debido a que estos metodos no son una instancia de la clase @overside es la palabra reservada para hacer el polimorfismo es aquí donde tanto podemos sobrescribir todo el metodo o agrega este cosas nuevas pero me diras y como hacemos esto

Collections  
  
Para este momento hemos aprendido mucho y es momento de poner ver nuevos conceptos es muy posible que ya algunos esten usando lo que me hemos mostrado a continuacion o ya tuviero su primer acercamiento con aquello sin mas empecemos.  
Que son las collections veamolo como mas bodegas para poder almacenar pero ya no datos. Ni cadenas string ni ni tipo de dato primitivo ahora almacenaremos Objetos y esto es gracias a nuestras clases warpper que son aquellas que nos permiten convertir y usar datos primitivos a objetos recalco las clases que veremos a continuacion cada una tiene caracteristicas que le hacen muy especial y diferente veamos.  
  


Como podemos visualizar en la imagen tenemos la superclase Collection de esta se despliega las clases hijas set y list como ya lo mencione antes cada una con sus caracteristicas pero comencemos con una.  
  
  
  
Set esta como caracteristicas principales no nos deja tener datos duplicados solo unicos, los objetos se almacenaran en desorden no contamos con indices esta cuenta con metodos para busca, eliminacion y agregar pero estos depedaran de cual de sus variantes usemos. Pero en general es muy eficiente para la eliminacion de datos, adicion y la busqueda recalco depende sus variantes pero cuales son estas veamos:  
  
  
🡪Hashset esta tiene como caracteristicas no podemos tener datos duplicados no tenemos un orden especificio pero esta a diferencia de su padre nos ayudara a encontrar datos o lo que tengamos almacenado de manera eficiente y rapida gracias a que esta asigna un dato numerico a cada objeto para poder buscarlo de forma rapida.  
  
🡪SortedSet este si ordena nuestro objetos o datos , no podemos tener datos repetidos ya que estos son eliminados al momento y en lo que respeta a datos y eficiencia rapida para busca y eliminacion es muy eficiente debido a que se usa tiempo algortimico pero aquí surge una incognita ¿Que es esto? El concepto general nos dice que este trabaja mucho mejor cuando la entrada es de muchos datos, ya que este aumenta de formas lenta a medida que mas datos se procesan y esto dado como resultado mas eficiencias, esto es debido a que la busqueda binaria separa nuestra lista, objetos o datos en dos para hacer una busqueda muchas rapida esto se hace simplemente con divisiones y mas eficiente cuando esta supera los 10 datos.  
  
🡪TreeSet esta nos ayuda ordenando nuestro datos de forma natural o de forma personalizada, no permite elementos duplicados si usted intenta agregar datos de este tipos simplemente seran eliminados, es basado en arboles binarios que quiere decir esto que este tipo de estructura de datos es muy eficiente en busca, eliminacion y insercion  
  
  
  
  
  
  
  
  
  
  
  
  
Por otro lado tenemos las list esta es un amor para mi es eficiente en cierta cosas como es ordenado y flexible con los datos cada elemento cuenta con un indice con el cual podemos acceder a este y estas nos permiten tener elementos duplicados sin tener algun inconveniente este nos ofrece variedad de metodos que veremos poco a poco sin mas que decir porfundicemos.  
  
  
🡪ArrayList estas nos permite tener elemento duplicados, para busquedas de elementos no es muye eficiente ya que esta es dinamica crece a medida que se le agregan sus elementos cada uno esta ordenado y cuenta con un indice, eliminar datos facil independientemente del metodo personalidada aunque recalco si eliminamos el elemento final es muy rapido pero elemento del medio puede ser muy costoso.  
  
Esta no es sincronizada algo que se me escapaba era el concepto de este que es esto. Cuando una lista cuenta con este topico se nos da conocer que es una lista segura aunque puede llegar a ser un poco lentos  
  
/\*  
\*futuro lector puedes completemetar este texto leyendo logica de programacion y haciendo tu propio proyecto   
\*/  
  
🡪Vector esta es una clase sincronizada debido a esta cualidad aquella es un poco lenta pero se mantienen seguro los datos que estan almacenados esto esta hecha para ser utilizada en entornos multiHilos pero aquí nace una duda que SON LOS HILOS? Estos son los encargados de las que tareas se cumplan pero como es posible esto pensemos en esto como un taxi o un delivery que se le carga ir del punto A al punto B es asi como los hilos nos ayudan a ejecutar datos pero entonces que son los MULTiHILOS? Estos son aquellos que se encargan de ejecutar varias tareas ojo no todas al mismo tiempo esto depende de tu procesador veamos un poco mas profundo para que logres entender esto.  
Tenemos dos tipos principales de hilos de usuario y de sistema el primero es creado por nosotros y gestionado por nuestra app o software. El segundo es gestionado y planificado por nuestro sistema este es muy pesado pero se beneficia de la optimizacion del sistema por otro lado el del usuario esto son ligeros, rapido de crear pero no se benefician del sistema que beneficios nos pueden dar los hilos es una excelente pregunta lector entre los principales ya antes mecionados tenemos fluidez, rapidez, eficiencia, mejor uso de recursos, modularidad y respuesta respuesta pero veamos conceptos claves para poder entender mas.  
  
\*- Concurencia : este nos dice que el sistema tiene la capacidad de gestionar multiples tareas a la vez pero se aclara que no se ejecutan al mismo tiempo pero puede progresar juntas se que esto parece una contradicion dicha en primera plana como ya antes se les menciona depende tambien de su multiprocesador.  
  
\*-Paralelismo : En este caso se nos dice que tiene la capacidad de ejecutar tareas multiples tareas al mimso tiempo ojo requiere multiprocesador ya antes mecionado.  
  
\*-DeadLock: situacion donde uno o mas hilos estan esperando que otros terminen por tal no avanzan.  
  
  
 🡪LinkedList esta es una clase que puede contener elementos duplicados ojo esta no esta sincronizada por es mas eficiente mas rapido se nos dice que esta clase es de doble fila que quiere decir esto nos dice que podremos navegar tanto para adelante y para atrás un pequeña analogia para que logres comprender digamos que quieres buscar la siguiente palabra “Fuck”. Y separamos la mitad y comenzamos a buscar en caso que no este vamos hasta adelante sino hasta atrás tenemos libertad de movimiento para poder buscar nuestros datos.   
Algo que acortar sobre esta lista cada numero sabe sin es el siguiente debido que estan en comunicación constante esto la convierten en una lista eficiente para eliminacion y bsuqueda de datos.  
  
  
  
  
  
  
  
  
Map  
  
  
  
si ponemos atencion directa a nuestro primero cuadro de herencia nos damos cuenta que los map no heredan de collections debido a que esto son un poco peculiares ya que esta representa una estructura de dato de mapeo y no de collecion simple de objeto pero que quiere decir esto es simple mira mi querido lector desde ahora cuando usted agrege datos en map se encontrada en su ide VALUE, KEY que quiere decir esto es simple cuando usted agrege un dato tambien tiene que agregarle un valor y podra ser llamado todo con este por otro lado se nos dice que las collections se agregaban datos simple debido que estos asi son ya que no tenian un key se que suena algo mas complejo pero creamen map es simple.  
  
Como ya sabemos de que y por que se diferencia profundicemos no trates de comprender todo esto en un dia suave recuerda bases solidas requieren tiempo y esfuerzo.  
  
  
  
🡪 Hash Map los elementos no se orden no se aceptan key duplicadas ni elementos nulos .  
  
🡪LinkedHashMap ordena los elementos conforme van ingresando esto provoca una lentitud en las busquedas de datos.  
  
  
🡪Tree map este orden los datos de forma natural es decir numeros de menor a mayor. Letras por orden alfabetico.  
  
  
  
  
  
  
  
  
  
  
  
  
 Clases abstractas   
  
Que son la estas clases me atrevo a llamarlas moldes estan no pertencen a las clases en si. Ni se puede declarar metodos dentro de estas pero si podemos dejar firmas a que me refiero con firmas quiero decir que podemos dejar declarado variables para poder ser sobreescribidas desde otra clase o lugar donde se use la clase abstracta es decir toda clase derivadas de esta deben sobreescribir estos metodo sin cuepro dado en la clase abstracta y darles cuerpo con multiples usos ya que como no se dice como debe implementar o usar tiene la libertad de hacerlo como sea mejor posible veamos unos cuantos ejemplos para poder comprender.

### Ejercicio 1: Instrumentos Musicales

1. **Crea una clase abstracta**InstrumentoMusical**con un método abstracto**tocar**.**
2. **Crea dos subclases concretas**Guitarra**y**Piano**que extiendan**InstrumentoMusical**y proporcionen una implementación para el método**tocar**.**

Class main{  
public static void main(String[] args){  
  
  
  
  
}  
  
}  
  
 Abstracta class InstrumentoMusical{  
 String nameInstrument;  
   
  
  
  
 abstracta void tocar(){  
  
}  
  
  
}  
 class Guiatarra extends InstrumentoMusical{  
  
  
 void tocar(){  
System.out.printltn(“estoy tocando una guitara escuchen pipipapa”);  
}  
  
  
}  
  
class Piano extends InstrumentoMusical{  
  
  
  
  
  
  
  
voidTocar(){  
  
System.out.println(“ estoy tocando un piano miren ajajjaa”);  
  
  
  
  
  
  
  
}  
}

1. **Crea una clase abstracta**AnimalZoologico**con un método abstracto**hacerSonido**.**
2. **Crea dos subclases concretas**Leon**y**Elefante**que extiendan**AnimalZoologico**y proporcionen una implementación para el método**hacerSonido**.**

class Main{  
public static void main(String[] args){  
  
  
  
AnimalZoologico leon = new leon();  
AnimalZoologico Elefante = new Elefante();  
  
  
leon.hacerSonido();  
Elefante.hacerSonido();  
  
}  
}  
  
abstract class AnimalZoologico{  
  
  
  
abstract void hacerSonido(){  
  
  
  
}  
}  
  
class Leon extends AnimalZoologico{  
  
  
  
 void hacerSonido(){  
  
System.out.println(“gruu gruuu”);  
}  
  
  
}  
  
class Elefante extends AnimalZoologico{  
  
  
  
void hacerSonido(){  
  
System.out.println(“fluu fluuu ”);  
}  
  
}

Miembros abstractos  
  
  
Miembros abstractos como ya venimos haciendo ejercicios de clases abstractas es mas que seguro que te diste cuenta de lo que tenemos metodos abstractos pues bueno los miembros abstractos engloba a todo su implementacion o su fin no es muy diferente al de sus familiares se deja un firma pero no se le da el contracto este se establece al extenderlo a una clase hija dependiendo de sus necesidades sin mas que decir acotar veamos ejemplos de estos para poder comprender y entender.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Clases anonimas   
  
  
Estas son efectivas para poder instanciar y sobreescribir datos desde un clase abastracta se que esto va encontra de lo que se dijo que no sea podia pero no es asi si se puede pero esto es gracias a las clasea anonimas ya que estan constan de una implementacion de una sola linea donde podemos sobrescribir estos clase y darle el comportamiento que deseamos pero ojo la vida de esta es muy limitada ya que no podremos reutilizarla ni llamarla ya que su alcance es minimo me atrevo a decir que es por tal que dicen que sirven para una sola cosa ya que no podremos hacer mas.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
cuando usar clases abstractas y interfaces.

1. **Crea una clase abstracta**FiguraGeometrica**con métodos abstractos**area**y**perimetro**.**
2. **Crea dos subclases concretas**Cuadrado**y**Circulo**que extiendan**FiguraGeometrica**y proporcionen una implementación para los métodos**area**y**perimetro**.**

class main{  
public static void main(String[] args){  
  
  
  
}  
}  
  
  
abstract class FiguraGeometrica{  
   
abstract double area();  
abstract double perimetro();  
}  
  
  
  
class Cuadrado extends FiguraGeometrica{  
  
Double medida;  
  
Cuadrado(double medida){  
 this.medida = medida;  
  
}  
  
Double area(){  
 double proceesMEdida = medidad \*meduda;  
  
 return processMEdida;  
  
  
}  
Double perimetro(){  
   
double premitroOne = 4 \* medida;  
 return premtroOne;  
  
}  
  
  
}

Abstract class datosOne{  
   
abstract datosTwo();  
abstract datosThree();  
  
 void datosLocus(); // listo como nos daremos cuenta este ultimo no cuenta con la palabra reservada “abstract” y por tal no estamos obligados a implimetarle a instanciarle   
  
}

Primero debemos comprender cuales son las diferencias entre ellos y sus características para esto debemos comprender muy pero muy bien su concepto general

Primero tenemos las clases abstracta estas nos dicen que servirán de models para la creación de otras clases debido a que estas solo tiene la firma y no el cuerpo y en las subclases poder darles este, no se pueden crear instancias y metodos y no podemos crear objeto, es decir si o si tenemos que heredarle estos datos a las clases hijas para poder continuar con el proceso

En las interfaces tenemos que estas cuentas con modificadores y son capaces de usar la herencia y no estan limitadas es decir mientras que las clases abstractas se comparten solo por medio de herencia estas se comparten entre familias

como elegir cual usar veamolo asi las clases Abstractas son aquellas que se apegan mas con los obejtos por otro lado tenemos las interfaces que son mas para acciones.  
  
Una buena practica es que el diseño de tus app este enfoca a interfacez y no a la impletacion concetrate en crear buenas asbtracciones y enfocate en la declaracion de tus metodos.  
  
  
las interfaces como ya antes mencione pueden heredar de otras usan lo que ya conocen la palabra extedns pero nos nace una duda esta compartiran sus datos en este caso sus metodos? Pero antes de responder la pregunta cuantos de ustedes se acuerdan de la herencia multiples y el problema de diamante pues para este punto se puede hacer con las interfaces sin ningun problema