TALLER 4

Integrantes

- Natalia Grijalba ngrijalbah@unal.edu.co
- David Fajardo dsfajardob@unal.edu.co
- Cristian Bernal crabernalmo@unal.edu.co
- Daniel Osuna dgosunar@unal.edu.co

1. Colecciones.
   a. La colección Set representa un conjunto de elementos. No admite duplicados. Tiene asociadas 3 clases a las que implemente: HashSet, LinkedHashSet y TreeSet.
   La colección Map es un conjunto de elementos con una clave o identificador. Tiene 4 clases a las que implementa: HashMap, Hashtale, LinkedHasMap y Tree Map.
   b. Una colección es un arreglo de objetos, al añadirle un valor de tipo primitivo a una colección dejaría de ser colección, seria únicamente un arreglo de programación.
   c.

ArrayList	HashMap	TreeMap	LinkedList
Representa una lista ordenada pero no clasificada. La interacción con esta colección es rápida.	No es ordenada, ni clasificada, permite tener claves con valor “null” y varios elementos con valor “null”	Es ordenada y clasificada.	Representa una lista ordenada y clasificada. La interacción con esta colección es más lenta que con ArrayList.

2.
   a. Se considera el método abstracto como un contrato ya que al incluirse en la clase base, este obliga a que todas las clases derivadas, lo sobrescriban con el mismo formato utilizado en la declaración.
   b. La clase abstracta permite crear métodos generales, que recrean un comportamiento común, pero sin especificación del funcionamiento.
   c. El polimorfismo consta de la capacidad que varias clases utilicen un método de forma diferente, en caso de un método abstracto facilita la implementación ya que no está definido concretamente.
   d.
      - - F - Todos los métodos de una clase abstracta tienen que ser abstractos.
      La clase abstracta es una clase que contiene al menos un método abstracto.
      - - V - Si una superclase declara un método abstracto una subclase debe implementar ese método.
      - - V - Un objeto de una clase que implementa una interfaz puede ser pensado como un objeto de ese tipo de interfaz.

e. Una clase abstracta es utilizada cuando se necesita definir tipos amplios de comportamiento en la raíz de la jerarquía de clases. La interfaz se utiliza cuando se requiere de modelar herencias múltiples, imponiendo conjuntos múltiples de comportamientos en la clase.

3. Características Estaticas
   a. Las utily class son clases que tienen métodos que su suelen utilizar muy a menudo, constan de atributos estáticos, estas clases deben ser públicas para poder ser accedidas desde cualquier parte, y también han de ser finales para que no puedan ser alteradas, sus métodos también tienen que ser estáticos dejándonos entonces con:
      -Clase principal Static Final
      -Atributos public static
      -Metodos public static
   b. Ejemplos
      *Para poder leer datos desde consola usamos Scanner para esto importamos java.util.Scanner;
      Uso:

```
Import java.util.Scanner;
Scanner sc = new Scanner (System.in) ;
Int pepito = sc.nextInt();
```

      *Para saber cual es el tamaño de un arreglo usamos uno de los metodos predefinidos de los arreglos el .lenght, con esto podemos recorrer arreglos sin sobre pasar su limite de posiciones
      Uso

```
String [] galletas = new String [42];
For(Int pepito=0; pepito<=galletas.lenght;pepito++){
//logica de lo que se supone haremos en el ciclo
}
```

      *Podemos usar la clase Math para determinar el valor absoluto de un numero siendo esto un ejemplo, muy Bueno para cuando el usuario quiere dañar el programa ingresando datos negativos, esta la importamos usando java.util.math;
      Uso:

```
Import java.util.Math;
Import java.util.Scanner;
Scanner sc= new Scanner(System.in);
System.out.println(“Por favor digite un numero positivo”);
Int pepito=sc. nextInt();
If(pepito<0){
Abs(pepito)
}
```

      Para demostrar un ejemplo de una utily class simple voy a crear una que me imprimirá una cadena de caracteres como un numero entero, flotante y doublé, útil por si queremos el

código aski de nuestros caracteres o simplemente hacer un casteo a alguno de estos tipos de dato.

```
public class EjemploUtility {

public static void main(String[] args) {
String s = "654.908000054";
float b = Float.parseFloat(s);
double c = Double.parseDouble(s);
int a = (int)b;

 System.out.println(a);
 System.out.println(b);
System.out.println(c);

}
}
```

4. El arrayList debe ser de tipo figuras, porque cada una de las figuras hereda de figura y se pueden generar a partir de esa clase; sino tienen que ser del tipo figura del cual se quiere hacer el arrayList porque el array solo es de una clase.