Universidad Nacional de Colombia

Periodo académico: 2015-2

Materia: Programación Orientada a Objetos

Jaime Enrique Ramírez Garzón

Juan Felipe Contreras Correa

1. Colecciones

a. Estos 2 tipos de colecciones tienen un par de diferencias, lo primero es que Set representa un conjunto de elementos, lo cuales no admiten duplicados, y tienen asociadas 3 clases a las que implemente: HashSet, LinkedHashSet y TreeSet, a cambio la colección Map es un conjunto de elementos con una clave o llave, el cual se divide en 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.

Tipo	ArrayList	HashMap	TreeMap	LinkedList
Características Especiales	Es una lista ordenada, pero no tiene clasificación, y tiene una ventaja de trabajo, ya que es muy rápida.	No tienen ningún orden, o clasificación, pero tiene una característica que los identifica, que es poder tener una clave y recibir datos nulos “null”.	Este tipo de listas se caracteriza por ser ordenada y clasificada.	Es una lista ordenada y clasificada. La interacción con esta colección es más lenta que con una ArrayList.

2.

a. Se considera el método abstracto como un contrato, por el motivo de que al crearse este, las subclases que hereden de ella deben usar y sobrescribir sus métodos.

b. La clase abstracta permite crear métodos generales, que recrean un comportamiento común, pero sin especificación del funcionamiento.

c. El polimorfismo hace que varias clases utilicen un método de forma diferente, ya que estas clases no tienen similitud.

d.

(F) Todos los métodos de una clase abstracta tienen que ser abstractos.

Esto es falso porque 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.

a. Las Utilily Class son clases que se usan muy a menudo, estas se conforman siempre de atributos estáticos, y deben ser públicas para poder ser accedidas desde cualquier parte del proyecto, también deben tener al inicio la palabra reservada Final, para que no puedan ser editada, y por ultimo sus métodos también tienen que ser estáticos.

b. Ejemplo de una Utility Class: Usamos la clase Math para hallar la raíz de un numero ingresado

```
Import java.util.Math;
Import java.util.Scanner;

Public class ClassMath{

   Scanner sc= new Scanner(System.in);
   System.out.println(“Por favor digite un número, al cual le sacaremos la raíz cuadrada: ”);
   Int num=sc. nextInt();
   Int aux;
    If (num => 0){
       aux = sqrt(num);
     }
   System.out.println(“La raiz cuadrada de “+num+” es igual a: “+aux);
}
```

Ahora vamos hacer un tipo de Utility Class, el cual nos halla fácilmente el área de un Triangulo:

```
import java.util.Scanner;
public class TestUtility {
    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double b,h;
      System.out.println("Ingrese la base del triangulo");
      b=sc.nextDouble();
      System.out.println("Ingresa la altura del triangulo");
      h=sc.nextDouble();
      double area;
      area=b*h/2;
     System.out.println(“Este es el área del triangulo: “+area);
    }
}
```

4. c. El ArrayList debe de ser del tipo “nombre de la clase padre” la cual en este caso es la abstracta, ya que es el ítem que tienen todos en común, porque son “parte a la vez del padre”.