TALLER 3

Integrantes:

-Héctor Augusto Daza Roa: -Hadazar@unal.edu.co

-Github: Hadazar.

-Julian Andres Ossa Castro: -Jaossacas@unal.edu.co

-Github:Jaossacas.

Repositorio:<https://github.com/Hadazar/Programacion-Orientada-a-Objetos-2015-2>

Desarrollo Taller:

1.

1. SET no tiene elementos repetidos, por lo que para un objeto en específico solo puede hacerse una referencia con él; MAP permite que existan dos o más claves para un mismo objeto, por lo que pueden hacerse múltiples referencias con él al objeto.
2. Cuando se almacena un dato de tipo primitivo en una colección, este ocupa un espacio dentro de la misma, y puede accederse a él haciendo referencia a su ubicación dentro de la colección.

|  |  |  |  |
| --- | --- | --- | --- |
| ArrayList | HashMap | TreeMap | ArrayDeque |
| Muy rápida accediendo a elementos, relativamente rápida agregando elementos si su capacidad inicial y de crecimiento están bien configuradas. | Es la implementación más genérica de Map. No garantiza el orden de las claves (de la misma forma que un HashSet). Si se requiere de un Map no-concurrente que no requiera ordenación de claves, este es adecuado. | Permite que sus claves puedan ser ordenadas, de la misma forma que *TreeSet.* Se usa en lugar de un HashMap solo si se necesita esta ordenación, ya que su rendimiento es mucho peor que el de HashMap en casi todas las operaciones (excepto iteración). | Es una implementación de Deque de rendimiento excepcional. Implementa tanto cola LIFO como FIFO al ser una Deque y es la cola que debería usarse si quiere implementarse una de estas dos estructuras. |

2.

1. Puede decirse que un método abstracto es un contrato, porque al crearlo tenemos la obligación de determinar su implementación en por lo menos una de las clases hijas a la clase a la que pertenece.
2. El propósito de la clase abstracta, es proveer una interfaz que permita hacer referencia a un método que poseen múltiples clases, pero que éstas implementan cada una de una forma diferente, permitiendo así el polimorfismo.
3. El polimorfismo, a través del método abstracto, nos da la ventaja de hacer referencia a un método que poseen múltiples clases con una única referencia, a pesar de que los métodos tengan implementaciones distintas en cada clase.

i. Todos los métodos de una clase abstracta deben ser declarados como abstractos.

(Falso, basta con que solo uno de los métodos de una clase sea abstracto para que

dicha clase sea abstracta).

ii. Si una superclase declara un método abstracto, una subclase debe implementar ese

método. (Falso, la subclase podría ser también una clase abstracta que no haga una implementación de este método).

iii. Un objeto de una clase que implementa una interfaz. (Verdadero).

1. Una clase abstracta se usa cuando se quiere que un método sea común para todas las clases hijas, a pesar de que cada una de éstas tengan una implementación diferente del mismo. Una interfaz se usa cuando múltiples clases aplican un mismo método que implementa cada una de forma diferente, pero en donde no tiene sentido que dichas clases hereden de una misma superclase.

3.

1. Para que una clase sea “Utility Class” todos sus métodos y atributos deben ser estáticos.