**HOJA INTERACTIVA DE APRENDIZAJE Nº 06**

**Apellidos y Nombres:** Casas Moreno Christian Joel

**Docente:** Nestor Audante Ramos **Fecha de entrega:** 30/10/2021

**Colecciones y manejo de excepciones**

1. **Indicadores:**
   1. Identifica uso de excepciones.
   2. Identifica las características y métodos de las colecciones.
   3. Identifica el uso de patrones de diseño creacionales.
   4. Reconoce la sintaxis de operaciones SQL.
2. **Actividades para desarrollar en forma individual:** 
   1. Complete la tabla sobre la sintaxis Java:

|  |  |
| --- | --- |
| **Código fuente** | **Errores** |
| public exception MiExcepcion{  private MiExcepcion(String mensaje){  super(mensaje);  }  } | * Nopuede haber un método de tipo exception, ni una clase exception. * El método Miexcepcion necesita de un tipo (void, int, …) para que pueda funcionar. * El super no debe tener parámetros de entrada. |
| Map<String, Integer> mapa = new HashMap();  mapa.add(“Perú”, 1523);  mapa.get(“Perú”); | * Se deben importar las bibliotecas respectivas. * Para mapas no se utiliza add, sino put. |
| Set<String> departamentos = new HashSet<>();  departamentos.add(“Loreto”);  departamentos.add(“Tacna”);  departamentos.add(“Puno”);  departamentos.get(3) | * Se deben importar las bibliotecas respectivas. * Para set no funciona el método get. |
| List<Integer> edades = new List<Integer>();  edades.put(12);  edades.put(12);  edades.get(0); | * Se deben importar las bibliotecas respectivas. * Al instanciar edades, no se puede poner new List<Integer>(); en todo caso se podría poner un ArrayList, un LinkedList, etc. * Para listas se utiliza add y no put. |

* 1. Complete la tabla escribiendo los conceptos aprendidos en clase

|  |
| --- |
| **Conceptos** |
| **Propagación de excepciones**  Es el mecanismo recomendado para interceptar errores que se produzcan durante la ejecución de las aplicaciones (divisiones por cero, lectura de archivos no disponibles, etc.) Básicamente, son objetos derivados de la clase System. |
| **Manejo de excepciones**  Agiliza el manejo de errores al permitir que el programa defina un bloque de código, llamado manejador de excepción, que se ejecuta automáticamente cuando ocurre un error. No es necesario verificar manualmente el éxito o fracaso de cada operación específica o llamada a un método. |
| **Métodos de Collection**  Son los métodos pertenecientes a la librería “Java Collection Framework”, que deriva del paquete estándar “java.util”. Las clases de la librería Collection sirven para almacenar colecciones de objetos, como pueden ser listas, mapas, sets o conjuntos, etc. |
| **List**  Es una colección que permite identificar los objetos independientemente de la implementación. Es decir, son genéricas. |
| **Set**  Es una coleccion que no puede contener elementos duplicados. |
| **Map**  Es una interface que nos permite representar una estructura de datos para almacenar pares "clave/valor"; de tal manera que para una clave solamente tenemos un valor. |
| **Implementaciones de las interfaces List, Set y Map**   * **ArrayList:** Permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica. * **LinkedList:** No permite posicionarse de manera absoluta (acceder directamente a un elemento de la lista) y por tanto no es conveniente para búsquedas, pero en cambio sí permite una rápida inserción al inicio/final de la lista. * **HashSet:** Es muy utilizada por los programadores web para la implementación de interfaces, la clase se conoce como interfaceSET y está contenida dentro de una clase hash que está construida en base a claves que permiten localizar objetos ya que da la posición del objeto en la tabla permitiendo así el acceso directo al elemento. * **TreeSet:** Está basada en el uso de una estructura de árbol permitiendo que los elementos estén ordenados bien por orden natural o bien por orden total definido por un Comparator. * **LinkedHashSet:** Los elementos del conjunto se encuentran en el orden que se insertan, similar a una lista, pero sin dejar ingresar valores repetido. * **HashMap:** Designa claves únicas para los valores correspondientes que se pueden recuperar en cualquier punto dado. * **LinkedHashMap:** Es una implementación LinkedList que es similar a una HashMap pero conserva el orden de inserción. * **TreeMap:** Es una implementación basada en árboles que puede almacenar datos clave-valor en orden clasificado de manera eficiente. |

* 1. Complete la tabla.

|  |
| --- |
| **Código fuente: Cree una clase con métodos sobrecargados donde se agregue y obtenga elementos de una lista, Set y Mapa.** |
| public class ListaSetMap {  public void agregarElementos(List<Integer> lista, Integer a){  lista.add(a);  }  public void obtenerElementos(List<Integer> lista, Integer b){  System.out.println("Elemento de posicion "+b+": "+lista.get(b));  System.out.println("Elementos de la lista: "+lista);  }  public void agregarElementos(Set<Integer> conjunto, Integer a){  conjunto.add(a);  }  public void obtenerElementos(Set<Integer> conjunto, Integer b){  System.out.println("Elementos del conjunto: "+conjunto);  }  public void agregarElementos(Map<String, Integer> mapa, String a, Integer b){  mapa.put(a, b);  }  public void obtenerElementos(Map<String, Integer> mapa, String c){  System.out.println("El valor perteneciente a "+c+" es: "+mapa.get(c));  System.out.println("Elementos del mapa: "+mapa);  }  public static void main(String[] args) {  ListaSetMap a = new ListaSetMap();  List<Integer> listaprueba = new ArrayList<>();  a.agregarElementos(listaprueba, 6);  a.agregarElementos(listaprueba, 9);  a.agregarElementos(listaprueba, 1);  a.obtenerElementos(listaprueba,1);  Set<Integer> conjuntoprueba = new HashSet<>();  a.agregarElementos(conjuntoprueba,7);  a.agregarElementos(conjuntoprueba,10);  a.agregarElementos(conjuntoprueba,0);  a.agregarElementos(conjuntoprueba,18);  a.obtenerElementos(conjuntoprueba,2);  Map<String, Integer> mapaprueba = new HashMap();  a.agregarElementos(mapaprueba,"Luis",18);  a.agregarElementos(mapaprueba,"Daniel",15);  a.agregarElementos(mapaprueba,"Riquelmer",2);  a.agregarElementos(mapaprueba,"Mario",15);  a.agregarElementos(mapaprueba,"Jhon",50);  a.agregarElementos(mapaprueba,"Jean",15);  a.obtenerElementos(mapaprueba,"Jhon");  }  } |

* 1. Caso de estudio: Interfaces

**Parte I**

Un estudiante de primaria le pide que escriba un programa para la distancia de dos objetos.

Utilice el patrón de diseño factory method para devolver la distancia en metros y en pies.

**Defina las clases necesarias, donde una sea clase abstracta y dos clases las implementaciones.**

**Parte II**

**Cree una tabla donde registrar las notas del curso. Adicionalmente ingrese cinco notas y luego modifique tres de ellas.**

* 1. Suba el documento resuelto y la solución del caso de estudio en un directorio llamado lab-6.

1. **Referencia de material** 
   1. Simón, Jose Antonio (2019). Introducción a las colecciones de java. Recuperado de <https://ukode.es/colecciones-i-introduccion-y-conceptos-basicos/>
   2. Formatel e-learning (2018). Colecciones. Recuperado de <http://formacion.desarrollando.net/cursosfiles/formacion/curso_754/java09.pdf>

1. **Conclusiones y recomendaciones**