Informe de Laboratorio 06

Tema: HashMap

Nota

Estudiante	Escuela	Asignatura
Roni Companocca Checco rcompanocca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: II Código: 20210558

Laboratorio	Tema	Duración
06	HashMap	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 29 Septiembre 2023	Al 30 Septiembre 2023

1. TAREA

1.1. Objetivos:

- Crear e inicializar HashMaps
- Realizar búsquedas secuencial y binaria en un HashMap
- Implementar métodos de ordenamiento en HashMap
- Solucionar problemas

1.2. Competencias a alcanzar:

- Diseña, responsablemente, sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
- Aplica de forma flexible, técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

2. EQUIPOS, MATERIALES Y TEMAS UTILIZADOS

- Sistema Operativo Windows
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.

3. URL DE REPOSITORIO GITHUB

- URL para el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO`
- URL para el laboratorio 06 en el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO/FP2-LAB6`

4. ACTIVIDADES

4.1. EJERCICIO RESUELTO

- - Creamos la Clase ColeccionesMap con los siguientes métodos que realizan las operaciones que se pueden realizar con el HashMap:

```java
// RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 06
// FUNDAMENTOS DE PROGRAMACION
// EJERCICIO PROPUESTO DESARROLLADO
import java.util.Map;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

public class ColeccionesMap{
    static Scanner consola = new Scanner(System.in);
    // interfaz implement
    Map diccionario1 = new HashMap();
    public void llenarHashMap(){
        // Insertar valores "key"-"value" al HashMap
        diccionario1.put("29458865","Maria");
        diccionario1.put("28784599","Jose");
        diccionario1.put("26558899","Luis");
        diccionario1.put("45785214","Juan");
        diccionario1.put("48889566","Mariela");
        diccionario1.put("48889544","Carlos");
        diccionario1.put("12389566","Rosa");
        diccionario1.put("48889540","Carlos");
        diccionario1.put("26558890","Luis");

        // definir iterator ara extraer o imprimir valores
        // recorremos y mostramos el diccionario
        for(Iterator iterador1 = diccionario1.keySet().iterator();
            iterador1.hasNext();){
```

```
            String key = (String)iterador1.next();
            String value = (String)diccionario1.get(key);
            System.out.println("DNI: "+key+" Alumno: "+value);
        }
        // mstramos el tamao del diccionario
        System.out.println("Tamao: "+diccionario1.size());
    }

    public void buscarEnHashMap(){
        // buscamos un valor en el diccionario
        System.out.println("Ingrese un Nombre a Buscar: ");
        String nombre = consola.next();
        if(diccionario1.containsValue(nombre)){
            System.out.println((nombre+" se encuentra en el Diccionario"));
        }else{
            System.out.println(nombre+ " No se encuentra!!");
        }
    }

    public void eliminarEnHashMap(){
        // eliminamos un valor en el diccionario
        System.out.println("Ingrese un Codigo a Eliminar: ");
        String llave = consola.next();
        diccionario1.remove( llave);

        for(Object key : diccionario1.keySet()){
            String llave2 = (String)key;
            String value2 = (String)diccionario1.get(llave2);
            System.out.println("DNI: "+llave2+" Alumno: "+value2);
        }
    }

    public void reemplazarEnHashMap(){
        // reemplazamos un valor en el diccionario
        System.out.println("Ingrese un codigo a reemplazar: ");
        String llave2 = consola.next();
        System.out.println("Ingrese el Nuevo Valor: ");
        String valorNuevo = consola.next();
        diccionario1.replace(llave2, valorNuevo);

        for(Object key3 : diccionario1.keySet()){
            String llave3 = (String)key3;
            String value3 = (String)diccionario1.get(llave3);
            System.out.println(("DNI: "+llave3+" Alumno: "+value3));
        }
    }
}
```

- Creamos la Clase Main, donde llamamos a nuestra clase y las operaciones implementadas:

```
public class Map01 {
    public static void main(String[] args){
        ColeccionesMap coleccion01 = new ColeccionesMap();
        System.out.println("----------------------");
        System.out.println("COLECCION HASHMAP");
        System.out.println("----------------------");
```

```
        coleccion01.llenarHashMap();
        coleccion01.buscarEnHashMap();
        coleccion01.reemplazarEnHashMap();
        coleccion01.eliminarEnHashMap();
    }
}
```

- Ejecucion: Ejecutamos el ejercicio desarrollado por la cual nos como respuesta lo siguiente.

```
----------------------
COLECCION HASHMAP
----------------------
DNI: 26558899 Alumno: Luis
DNI: 48889540 Alumno: Carlos
DNI: 28784599 Alumno: Jose
DNI: 29458865 Alumno: Maria
DNI: 48889566 Alumno: Mariela
DNI: 48889544 Alumno: Carlos
DNI: 12389566 Alumno: Rosa
DNI: 26558890 Alumno: Luis
DNI: 45785214 Alumno: Juan
 Tamao : 9
Ingrese un Nombre a Buscar:
Carlos
Carlos se encuentra en el Diccionario
Ingrese un codigo a reemplazar:
48889544
Ingrese el Nuevo Valor:
Roni
DNI: 26558899 Alumno: Luis
DNI: 48889540 Alumno: Carlos
DNI: 28784599 Alumno: Jose
DNI: 29458865 Alumno: Maria
DNI: 48889566 Alumno: Mariela
DNI: 48889544 Alumno: Roni
DNI: 12389566 Alumno: Rosa
DNI: 26558890 Alumno: Luis
DNI: 45785214 Alumno: Juan
Ingrese un Codigo a Eliminar:
48889544
DNI: 26558899 Alumno: Luis
DNI: 48889540 Alumno: Carlos
DNI: 28784599 Alumno: Jose
DNI: 29458865 Alumno: Maria
DNI: 48889566 Alumno: Mariela
DNI: 12389566 Alumno: Rosa
DNI: 26558890 Alumno: Luis
DNI: 45785214 Alumno: Juan
```

4.2. EJERCICIO PROPUESTO

- Cree un Proyecto llamado Laboratorio6
- Usted deberá crear las dos clases Soldado.java y VideoJuego5.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.

- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para crear el tablero utilice la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (usar HashMaps). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento (indicar conclusiones respecto a este ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como programa iterativo.
- la clase VideoJuego5.java

```java
// RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 06 - HASHMAP
// FUNDAMENTOS DE PROGRAMACION
import java.util.*;

public class VideoJuego5 {

    public static void main(String[] args) {

        // DECLARACIN DE VARIABLES Y ESTRUCTURAS DE DATOS NECESARIAS
        HashMap<String, Soldado> ejercito1 = new HashMap<>();
        HashMap<String, Soldado> ejercito2 = new HashMap<>();
        HashMap<String, Soldado> tablero = new HashMap<>();
        int batallon1, batallon2;
        int vidatotal1 = 0, vidatotal2 = 0;
        double promedioVida1 = 0, promedioVida2 = 0;

        // BUCLE PARA DESIGNAR LA CANTIDAD DE FILAS Y COLUMNAS DEL TABLERO
        for (int i = 1; i <= 10; i++) {
            for (int j = 1; j <= 10; j++) {
                String posicion = i + "x" + j;
                tablero.put(posicion, new Soldado());
            }
        }

        // CREACIN DEL NMERO DE POSICIONES DE CADA EJRCITO
        batallon1 = aleatorio(1, 10);
        batallon2 = aleatorio(1, 10);

        // INICIALIZAR MAPAS DE EJRCITOS
        inicializarEjercito(ejercito1, batallon1);
        inicializarEjercito(ejercito2, batallon2);

        // GENERAR EJRCITOS VLIDOS
        generarEjercitos(ejercito1, ejercito2, tablero);
```

```
// IMPRIMIR EL TABLERO
imprimirTablero(tablero);

// IMPRIMIR LOS SOLDADOS DE MAYOR VIDA DE CADA EJRCITO
System.out.println("Soldado de mayor vida del ejrcito 1");
SoldadoConMayorVida(ejercito1);
System.out.println("Soldado de mayor vida del ejrcito 2");
SoldadoConMayorVida(ejercito2);

// IMPRIMIR LA VIDA TOTAL Y EL PROMEDIO DEL EJRCITO 1
System.out.println("\nEJRCITO 1: ");
for (Soldado soldado : ejercito1.values()) {
    vidatotal1 += soldado.getPuntos();
}
promedioVida1 = vidatotal1 / (double) ejercito1.size();
System.out.println("Vida total: " + vidatotal1);
System.out.println("Promedio de vida: " + promedioVida1);

// IMPRIMIR LA VIDA TOTAL Y EL PROMEDIO DEL EJRCITO 2
System.out.println("\nEJRCITO 2: ");
for (Soldado soldado : ejercito2.values()) {
    vidatotal2 += soldado.getPuntos();
}
promedioVida2 = vidatotal2 / (double) ejercito2.size();
System.out.println("Vida total: " + vidatotal2);
System.out.println("Promedio de vida: " + promedioVida2);

// IMPRIMIR LOS SOLDADOS CREADOS EN EL ORDEN POR DEFECTO
System.out.println("\nLista ejrcito 1:");
for (Soldado soldado : ejercito1.values()) {
    imprimir(soldado);
}
System.out.println("\nLista ejrcito 2:");
for (Soldado soldado : ejercito2.values()) {
    imprimir(soldado);
}

// IMPRIMIR LOS DATOS DE LOS SOLDADOS ORDENADOS DE MAYOR A MENOR
    DEPENDIENDO DE SU NIVEL DE VIDA
ArrayList<Soldado> listaSoldados1 = new ArrayList<>(ejercito1.values());
ArrayList<Soldado> listaSoldados2 = new ArrayList<>(ejercito2.values());
ordenarPorVidaMetodoA(ejercito1);
ordenarPorVidaMetodoB(ejercito2);
System.out.println("\nEjrcito 1 Ordenados por nivel de vida");
for (Soldado soldado : listaSoldados1) {
    imprimir(soldado);
}
System.out.println("\nEjrcito 2 Ordenados por nivel de vida");
for (Soldado soldado : listaSoldados2) {
    imprimir(soldado);
}

// MOSTRAR EJRCITO GANADOR LA MTRICA USADA PARA DESIGNAR AL GANADOR ES EL
    PROMEDIO DEL NIVEL DE VIDA DE CADA EJRCITO
if (promedioVida1 > promedioVida2) {
```

```
            System.out.println("\nGANADOR ***EJRCITO 1***");
        } else if (promedioVida1 < promedioVida2) {
            System.out.println("\nGANADOR ***EJRCITO 2***");
        } else {
            System.out.print("\n***ES UN EMPATE***");
        }
    }

    // METODO PARA CREAR NUMEROS ALEATORIOS EN UN RANGO
    public static int aleatorio(int min, int max) {
      return(int)(Math.random()*(max-min+1)+min);
    }

    // METODO PARA INICIAR UN EJRCITO
    public static void inicializarEjercito(HashMap<String, Soldado> ejercito, int
        num) {
        for (int i = 0; i < num; i++) {
            ejercito.put("Soldado" + i, new Soldado());
        }
    }

    // METODO PARA GENERAR DATOS DEL OBJETO SOLDADO
    public static Soldado generarDatos() {
      Soldado soldadito = new Soldado();
      soldadito.setPuntos(aleatorio(1,5));
      soldadito.setColumna(aleatorio(1,10));
      soldadito.setFila(aleatorio(1,10));
      return soldadito;
    }

    // METODOS PARA GENERAR LOS EJERCITOS DE MANERA ALEATORIA
    public static void generarEjercitos(HashMap<String, Soldado> ejercito1,
        HashMap<String, Soldado> ejercito2, HashMap<String, Soldado> tablero) {
        ArrayList<String> posicionesOcupadas = new ArrayList<>();

        // Generar los soldados y asegurarse de que sus posiciones sean nicas
        for (int i = 0; i < ejercito1.size() + ejercito2.size(); i++) {
            Soldado soldado = generarDatos();
            String posicion = soldado.getFila() + "x" + soldado.getColumna();

            // Verificar que la posicin no est ocupada
            while (posicionesOcupadas.contains(posicion)) {
                soldado = generarDatos();
                posicion = soldado.getFila() + "x" + soldado.getColumna();
            }

            posicionesOcupadas.add(posicion);

            // Asignar soldado a ejrcito 1 o ejrcito 2
            if (i < ejercito1.size()) {
                ejercito1.put("Soldado" + i + "x1", soldado);
            } else {
                ejercito2.put("Soldado" + (i - ejercito1.size()) + "x2", soldado);
            }

            // Asignar soldado al tablero
```

```
            tablero.put(posicion, soldado);
        }

        // Actualizar las columnas de los soldados en ejrcito 1 y ejrcito 2
        for (String clave : ejercito1.keySet()) {
            Soldado soldado = ejercito1.get(clave);
            soldado.setColumn(soldado.getPuntos() + "[E1]");
        }
        for (String clave : ejercito2.keySet()) {
            Soldado soldado = ejercito2.get(clave);
            soldado.setColumn(soldado.getPuntos() + "[E2]");
        }
    }

    // METODO PARA AADIR LOS EJRCITOS AL TABLERO
    public static void aadirTablero(HashMap<String, Soldado> ejercito,
         HashMap<String, Soldado> tablero) {
        for (String posicion : ejercito.keySet()) {
            tablero.put(posicion, ejercito.get(posicion));
        }
    }

    // METODO PARA IMPRIMIR EL TABLERO EN LA CUAL SE DESARROLLA EL JUEGO
    public static void imprimirTablero(HashMap<String, Soldado> tablero) {
        System.out.println("\tA\tB\tC\tD\tF\tG\tH\tI\tJ");
        for (int i = 1; i <= 10; i++) {
            System.out.print(i);
            for (int j = 1; j <= 10; j++) {
                String posicion = i + "x" + j;
                Soldado soldado = tablero.get(posicion);
                System.out.print("\t" + soldado.getColumn());
            }
            System.out.println("\n");
        }
    }

    //METODO PARA IMPRIMIR LOS SOLDADOS DE MAYOR VIDA
    public static void SoldadoConMayorVida(HashMap<String, Soldado> soldados) {
        Soldado mayor = null;

        for (Soldado soldado : soldados.values()) {
            if (mayor == null || soldado.getPuntos() > mayor.getPuntos()) {
                mayor = soldado;
            }
        }

        if (mayor != null) {
            imprimir(mayor);
        } else {
            System.out.println("No se encontraron soldados.");
        }
    }

    // METODO PARA IMPRIMIR EL NOMBRE, LA POSICION Y NIVEL DE VIDA DEL SOLDADO
    public static void imprimir(Soldado soldadito) {
       System.out.println("Nombre: "+soldadito.getNombre()+"\nPosicion:
```

```
            "+soldadito.getColumna()+"X"+soldadito.getFila()+"\tVida:
            "+soldadito.getPuntos());
    }

    // METODO QUE NOS AYUDA A ORDENAR LOS SOLDADOS DE ACUERDO A SU NIVEL DE VIDA,
        USUANDO UN ALGORITMO DE ORDENAMIENTO DE BURBUJA
    public static void ordenarPorVidaMetodoA(HashMap<String, Soldado> soldados) {
        // Obtener los valores (los soldados) del HashMap y almacenarlos en una
            lista
        List<Soldado> listaSoldados = new ArrayList<>(soldados.values());

        Soldado aux = new Soldado();
        for (int i = 0; i < listaSoldados.size() - 1; i++) {
            for (int j = 0; j < listaSoldados.size() - i - 1; j++) {
                if (listaSoldados.get(j).getPuntos() < listaSoldados.get(j +
                    1).getPuntos()) {
                    aux = listaSoldados.get(j);
                    listaSoldados.set(j, listaSoldados.get(j + 1));
                    listaSoldados.set(j + 1, aux);
                }
            }
        }

        // Actualizar el HashMap con los soldados ordenados
        int index = 0;
        for (String clave : soldados.keySet()) {
            soldados.put(clave, listaSoldados.get(index));
            index++;
        }
    }

    // METODO QUE NOS AYUDA A ORDENAR LOS SOLDADOS DE ACUERDO A SU NIVEL DE VIDA,
        EN ESTA OCACION DIFERENTE A LA ANTERIOR QUE ERA ALGORITMO DE BURBUJA
    public static void ordenarPorVidaMetodoB(HashMap<String, Soldado> soldados) {
        // Obtener los valores (los soldados) del HashMap y almacenarlos en una
            lista
        List<Soldado> listaSoldados = new ArrayList<>(soldados.values());

        // Ordenar la lista en orden descendente por puntos de vida
        Collections.sort(listaSoldados, new Comparator<Soldado>() {
            public int compare(Soldado s1, Soldado s2) {
                // Orden descendente por puntos de vida
                return Integer.compare(s2.getPuntos(), s1.getPuntos());
            }
        });

        // Actualizar el HashMap con los soldados ordenados
        int index = 0;
        for (String clave : soldados.keySet()) {
            soldados.put(clave, listaSoldados.get(index));
            index++;
        }
    }
}
```

- la clase Soldado.java

```java
// RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 06
// FUNDAMENTOS DE PROGRAMACION - LABORATORIO
public class Soldado {
   private String nombre;
   private int fila;
   private int columna;
   private int puntos;
   private String column;

   public Soldado() {
      nombre = "";
      fila = 0;
      columna = 0;
      puntos = 0;
      column = "";
   }

   // METODOS MUTADORES
   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public void setFila(int fila) {
      this.fila = fila;
   }

   public void setColumna(int columna) {
      this.columna = columna;
   }

   public void setPuntos(int puntos) {
      this.puntos = puntos;
   }

   public void setColumn(String column) {
      this.column = column;
   }

   // METODOS ACCESORES
   public String getNombre() {
      return nombre;
   }

   public int getFila() {
      return fila;
   }

   public int getColumna() {
      return columna;
   }

   public int getPuntos() {
      return puntos;
   }
```

```
   public String getColumn() {
      return column;
   }
}
```

- Ejecucion

```
       A       B       C       D       F       G       H       I       J
1                      4[E2]   4[E2]

2      1[E1]                                           3[E2]

3              5[E2]   1[E1]                                           5[E1]

4      3[E2]

5                      2[E1]                   5[E1]

6      1[E2]   5[E2]

7

8                                                                      1[E1]

9                      4[E2]

10             4[E2]                   5[E1]   1[E2]

Soldado de mayor vida del ejercito 1
Nombre: Soldado0x1
Posicion: 10X5 Vida: 5
soldado de mayor vida del ejercito 2
Nombre: Soldado1x2
Posicion: 6X2  Vida: 5

EJERCITO 1:
Vida total: 20
Promedio de vida: 2.857142857142857

EJERCITO 2:
Vida total: 34
Promedio de vida: 3.4

Lista ejercito 1:
Nombre: Soldado0x1
Posicion: 10X5 Vida: 5
Nombre: Soldado1x1
Posicion: 8X9  Vida: 1
Nombre: Soldado2x1
Posicion: 3X3  Vida: 1
Nombre: Soldado3x1
Posicion: 5X6  Vida: 5
Nombre: Soldado4x1
Posicion: 5X3  Vida: 2
Nombre: Soldado5x1
```

```
Posicion: 2X1  Vida: 1
Nombre: Soldado6x1
Posicion: 3X9  Vida: 5

Lista ejercito 2:
Nombre: Soldado0x2
Posicion: 1X4  Vida: 4
Nombre: Soldado1x2
Posicion: 6X2  Vida: 5
Nombre: Soldado2x2
Posicion: 10X6 Vida: 1
Nombre: Soldado3x2
Posicion: 9X3  Vida: 4
Nombre: Soldado4x2
Posicion: 2X7  Vida: 3
Nombre: Soldado5x2
Posicion: 4X1  Vida: 3
Nombre: Soldado6x2
Posicion: 10X2 Vida: 4
Nombre: Soldado7x2
Posicion: 6X1  Vida: 1
Nombre: Soldado8x2
Posicion: 3X2  Vida: 5
Nombre: Soldado9x2
Posicion: 1X3  Vida: 4

Ejercito 1 Ordenados por nivel de vida
Nombre: Soldado0x1
Posicion: 10X5 Vida: 5
Nombre: Soldado3x1
Posicion: 5X6  Vida: 5
Nombre: Soldado6x1
Posicion: 3X9  Vida: 5
Nombre: Soldado4x1
Posicion: 5X3  Vida: 2
Nombre: Soldado1x1
Posicion: 8X9  Vida: 1
Nombre: Soldado2x1
Posicion: 3X3  Vida: 1
Nombre: Soldado5x1
Posicion: 2X1  Vida: 1

Ejercito 2 Ordenados por nivel de vida
Nombre: Soldado1x2
Posicion: 6X2  Vida: 5
Nombre: Soldado8x2
Posicion: 3X2  Vida: 5
Nombre: Soldado0x2
Posicion: 1X4  Vida: 4
Nombre: Soldado3x2
Posicion: 9X3  Vida: 4
Nombre: Soldado6x2
Posicion: 10X2 Vida: 4
Nombre: Soldado9x2
Posicion: 1X3  Vida: 4
Nombre: Soldado4x2
```

```
Posicion: 2X7  Vida: 3
Nombre: Soldado5x2
Posicion: 4X1  Vida: 3
Nombre: Soldado2x2
Posicion: 10X6 Vida: 1
Nombre: Soldado7x2
Posicion: 6X1  Vida: 1

GANADOR ***EJERCITO 2***
```

5. CUESTIONARIO

5.1. ¿Cómo se declara e inicializa un HashMap?

- Para declarar e inicializar un HashMap en Java

```
import java.util.HashMap;
HashMap<KeyType, ValueType> nombreDelMapa = new HashMap<>();
```

5.2. ¿Qué ventajas tienen los HashMap con respecto a los arreglos y ArrayList?

- Los HashMap en Java y otras estructuras de datos basadas en tablas de hash ofrecen varias ventajas en comparación con los arreglos (arrays) y ArrayLists como: Búsqueda eficiente, flexibilidad en las claves, capacidad dinámica, eliminación eficiente, no hay restricciones en la ubicación, claves únicas, etc.

5.3. ¿Mencione 4 métodos importantes del HashMap? ¿Qué finalidad tienen?

- put(Key key, Value value): Agregar un par clave-valor al HashMap. Si la clave ya existe en el mapa, el valor asociado se actualiza con el nuevo valor proporcionado. Si la clave no existe, se crea un nuevo par clave-valor en el mapa.
- get(Object key): Recuperar el valor asociado con una clave específica en el HashMap.
- remove(Object key): Eliminar el par clave-valor asociado con la clave proporcionada del HashMap.
- containsKey(Object key): Verificar si el HashMap contiene una clave específica. Devuelve true si la clave existe en el mapa y false en caso contrario.

6. REFERENCIAS

- M. Aedo, “Fundamentos de Programación 2 - Tópicos de Programación Orientada a Objetos”, Primera Edición, 2021, Editorial UNSA.
- `https://github.com/rescobedoq/programacion.git`
- J. Dean, Ïntroduction to programming with Java: A Problem Solving Approach”, Third Edition, 2021, McGraw-Hill.
- C. T. Wu, .An Introduction to Object-Oriented Programming with Java”, Fifth Edition, 2010, McGraw-Hill.
- P. Deitel, ”Java How to Program”, Eleventh Edition, 2017, Prentice Hall.