1.- Ejercicio de listas:

a) Escribe una clase Persona que tenga 2 propiedades: nombre y edad e sobreescribe el método toString().
b) Crea un ArrayList de Persona.
c) Añade 5 personas.
d) Recorre la lista utilizando un Iterator, mostrando las personas incluidas en ella.
e) Recorre la lista utilizando un bucle for, mostrando las personas incluidas en ella.
f) Recorre la lista utilizando un bucle for-each.
g) Crea una nueva Persona "amigo" y añádela al principio de la lista.
h) Añade de nuevo al "amigo" en la posición 2.
i) Averigua como comprobar si la lista contiene el objeto "amigo" (Existe un método).
j) Imprime la Persona en la 4 posición.
k) Actualiza el nombre de la persona en la posición 3.
l) Elimina la Persona en la posición 2.
m) Escribe un método que permita buscar una persona por su nombre en la lista.
n) Escribe un método que devuelva la lista inversa.
o) Escribe un método que sirva para comparar si 2 listas son iguales.
p) Escribe un método que copie todos los elementos de una lista en otra.
q) Haz que Persona implemente la interface Comparable para establecer que el orden natural entre personas sea por nombre. Ejecuta Collections.sort sobre la lista e imprímela para comprobar qué ha sucedido.
r) Escribe un Comparator para ordenar la lista por edad.
s) Escribe un Comparator para ordenar la lista por nombre y edad.

1.2.- Haz que Persona implemente la interface Comparable para establecer que el orden natural entre personas sea por nombre. Ejecuta Collections.sort sobre la lista e imprímela para comprobar qué ha sucedido.

1.3.- Crear un Comparator que nos permita ordenar la lista anterior por edad. ¿Qué cambios hay que hacer para que ordene descendentemente?

1.4.- Crear un Comparator que nos permita ordenar la lista anterior por nombre y edad.

4.- Ejercicio de colas:

a) Crea un ArrayDeque de Persona.
b) Añadir 5 personas con "add".
c) Crear un método para imprimir la cola (while – pop).
d) Añadir de nuevo las 5 personas pero utilizando el método "push".
e) Imprimir de nuevo la cola.
f) ¿Qué diferencia se observa?

5.- Cola con prioridad:

a) Crea una clase Paciente que herede de Persona y que tenga una propiedad “int urgencia”.
b) Crea una PriorityQueue y añádele 5 pacientes con diferentes valores en “urgencia”.
c) Recorre la cola mostrando sus elementos (while – remove). ¿Qué orden observas? ¿Porqué es así?
d) Asegúrate de que 3 pacientes de diferentes edades tengan la misma urgencia y crea un nuevo comparador que sirva para ordenar los pacientes por urgencia (desc) y edad (asc) y utilízalo en el constructor de la PriorityQueue. Imprime de nuevo la cola ¿Qué observamos?

6.- Ejercicio de conjuntos

a) Crea un HashSet de la clase Equipo cuyas propiedades son nombre y puntos y añade 4 equipos al conjunto.
b) Realizar un método para imprimir el conjunto que utilice un Iterator. Imprime el conjunto. ¿Qué observas respecto al orden de inserción y de impresión?
c) Crea un nuevo Equipo y añádelo 2 veces al conjunto. Vuelve a imprimir el conjunto. ¿Qué ocurre?
d) Crea un método “initList” que devuelva una lista de 4 Equipos en un ArrayList y refactoriza la aplicación para que el HashSet utilice esta lista en su constructor.
e) Crea un LinkedHashSet utilizando en su constructor el mismo método que creaste en d). Imprime el LinkedHashSet. ¿Qué observas respecto al orden de inserción y de impresión?
f) Crea un TreeSet a partir de la lista devuelta en el método “initList”. Imprime el TreeSet y ejecuta. ¿Qué ocurre?
g) Haz que la clase Equipo implemente Comparable y ordene por nombre. Vuelve a ejecutar.
h) Crea un Comparator de Equipo que ordene por los puntos y utilízalo en el constructor del TreeSet. Imprime el conjunto.
i) Modifica el comparador anterior para que el usuario pueda indicarle si quiere que ordene ascendentemente o descendentemente.

7.- Ejercicio de mapas:

a) Añade a la clase Persona el campo “DNI”.
b) Crea un HashMap y guarda en ella las personas devueltas por el método initList utilizando el valor del campo DNI como llave.
c) Escoge uno de los DNIs al azar y recupera esa persona desde el mapa.
d) Recorre la lista de llaves utilizando un Iterator, imprimiendo sus valores.

8.- Crea una nueva lista de enteros. Genera 100 números enteros y añádelos a la lista.

a) Halla la suma de los números recorriedo la lista con un Iterator.
b) Halla el valor máximo recorriendo la lista con un bucle for.
c) Halla el valor mínimo recorriendo la lista con un bucle for-each.

9.- Qué interface representa una colección que no admite elementos duplicados?

10.- Qué interface se considera la raíz de las colecciones?

11.- Escribe un programa que dada una frase de entrada, almacene, sin repetirlas, todas las letras que aparecen en la frase. Muéstralas por pantalla.

12.- Qué interface representa una colección que mantiene los elementos hasta que son procesados?

13.- Qué interface representa una colección que mapea llaves y valores?

14.- Cómo se llama la interface que representa una cola “double ended”?

15.- Escribe un programa que dada una frase de entrada, almacene cuantas veces aparece cada letra. Imprime las letras que aparecen en la frase indicando el número de veces que aparece cada letra.

16.- Las principales interfaces dentro de las Java Collections están organizadas en 2 árboles distintos. En particular una interface no está considerada como una verdadera Collection y por eso tiene un árbol de herencia propio. Cúal es el nombre de esta interface?

17.- Qué interface representa una colección ordenada que puede contener duplicados?

18.- Menciona 3 métodos de iterar sobre los elementos de una lista.

19.- Cuál será la salida del siguiente programa?

```
public class priorityQueue
{
    public static void main(String[] args)
    {
        PriorityQueue<Integer> queue =
                        new PriorityQueue<>();
        queue.add(11);
        queue.add(10);
        queue.add(22);
        queue.add(5);
        queue.add(12);
        queue.add(2);

        while (queue.isEmpty() == false)
            System.out.printf("%d ", queue.remove());

        System.out.println("\n");
```

```
    }
}
```

a) 11 10 22 5 12 2
b) 2 12 5 22 10 11
c) 2 5 10 11 12 22
d) 22 12 11 10 5 2

20.- Escribe un método que reciba un ArrayList de String y devuelva la longitud de la cadena más larga.

21.- Cuál será la salida del siguiente programa?

```
public class Treeset
{
    public static void main(String[] args)
    {
        TreeSet<String> treeSet = new TreeSet<>();

        treeSet.add("Geeks");
        treeSet.add("for");
        treeSet.add("Geeks");
        treeSet.add("GeeksforGeeks");

        for (String temp : treeSet)
            System.out.printf(temp + " ");

        System.out.println("\n");
    }
}
```

a) Geeks for Geeks GeeksforGeeks
b) Geeks for GeeksforGeeks
c) Geeks GeeksforGeeks for
d) for GeeksforGeeks Geeks

22.- Cuál será la salida del siguiente programa?

```
public class linkedList
{
    public static void main(String[] args)
    {
        List<String> list1 = new LinkedList<>();
        list1.add("Geeks");
        list1.add("for");
        list1.add("Geeks");
        list1.add("GFG");
        list1.add("GeeksforGeeks");
```

```
            List<String> list2 = new LinkedList<>();
            list2.add("Geeks");

            list1.removeAll(list2);

            for (String temp : list1)
                System.out.printf(temp + " ");

            System.out.println();
        }
    }
```

a) for Geeks GFG GeeksforGeeks
b) for GFG Geeks GeeksforGeeks
c) for GFG for
d) for GFG GeeksforGeeks

23.- Cuál será la salida del siguiente programa?

```
public class hashSet
{
    public static void main(String[] args)
    {
        HashSet<String> hashSet = new HashSet<>();
        hashSet.add("Geeks");
        hashSet.add("for");
        hashSet.add("Geeks");
        hashSet.add("GeeksforGeeks");

        System.out.println(hashSet);

    }
}
```

a) [Geeks, for, Geeks, GeeksforGeeks]
b) [GeeksforGeeks, Geeks, for]

24.- Escribe un método que reciba una lista de palabras y devuelva una lista sin palabras duplicadas.

25.- Cuál será la salida del siguiente programa?

```
    public class stack
    {
        public static void main(String[] args)
        {
            List<String> list = new LinkedList<>();
            list.add("Geeks");
```

```
        list.add("for");
        list.add("Geeks");
        list.add("GeeksforGeeks");
        Iterator<Integer> iter = list.iterator();

        while (iter.hasNext())
           System.out.printf(iter.next() + " ");

        System.out.println();
     }
}
```

a) Geeks for Geeks GeeksforGeeks
b) GeeksforGeeks Geeks for Geeks
c) Runtime Error
d) Compilation Error

26.- Cuál será la salida del siguiente programa?

```
class Demo {
        public void show() {
                ArrayList<String> list = new ArrayList<String>();
                ArrayList<Integer> list1 = new ArrayList<Integer>();

                boolean check = (list.getClass() == list1.getClass());
                System.out.println(check);
        }
        }

        public class Main {
                public static void main(String[] args) {
                        Demo demo = new Demo();
                        demo.show();
                }
        }
```

A. true
B. false

27.- Cuál será la salida del siguiente programa?

```
        class Demo {
        public void show()
           {
              List<Integer> list = new LinkedList<Integer>();
              list.add(1);
              list.add(4);
              list.add(7);
              list.add(5);
              Collections.sort(list); // line 9
```

```
        System.out.println(list);
    }
} public class Main {
public static void main(String[] args)
    {
        Demo demo = new Demo();
        demo.show();
    }
}
```

A. Compilation Error at line 9
B. [1, 4, 5, 7]
C. [1, 4, 7, 5]

28.- Escribe un método que reciba 2 listas de enteros aleatorios, encuentre todos los enteros que están en simultáneamente en ambas listas y devuelva todos los elementos encontrados (sin repetirlos).
a) Sólo puedes utilizar bucles for-each o Iterator para buscar en las listas de aleatorios.
b) Utiliza algún método de las Collection para agilizar la búsqueda.

29.- Escribe un método "mapIntersection" que reciba 2 mapas de String a Integer y devuelva la instersección (los elementos comunes) entre ambos mapas. Se considera un elemento común aquel que existe en los dos mapas y tiene la misma llave asociada al mismo valor.

30.- Escribe un método "maxOccurrences" que acepte una lista de enteros y devuelva el elemento más repetido y el número de veces que aparece.

Generics

1.- Escribe un método genérico para intercambiar la posición de 2 elementos en un array.

2.- Despúes del borrado de tipo, como será esta clase:

```
public class Pair<K, V> {

    public Pair(K key, V value) {
        this.key = key;
        this.value = value;
    }

    public K getKey(); { return key; }
    public V getValue(); { return value; }
```

```
    public void setKey(K key)     { this.key = key; }
    public void setValue(V value) { this.value = value; }

    private K key;
    private V value;
}
```

3.- Despúes del borrado de tipo, como será esta método:

```
public static <T extends Comparable> int findFirstGreaterThan(T[ ] at, T elem) {
    // ...
}
```

4.- El siguiente método compila correctamente? Si no compila, porqué?

```
public static <T extends Number> void print(List<T> list) {
    for (Number n : list)
        System.out.print(n + " ");
    System.out.println();
}
```

5.- Compilará correctamente la siguiente clase? Si no compila, porqué?

```
public class Singleton<T> {

    public static T getInstance() {
        if (instance == null)
            instance = new Singleton<T>();

        return instance;
    }

    private static T instance = null;
}
```

6.- Dada la siguiente clase:

```
class Shape { /* ... */ }
class Circle extends Shape { /* ... */ }
class Rectangle extends Shape { /* ... */ }

class Node<T> { /* ... */ }
```

Compilará el siguiente código? Si no compila, porqué?

```
Node<Circle> nc = new Node<>();
Node<Shape>  ns = nc;
```

7.- Dada la siguiente clase:

```
class Node<T> implements Comparable<T> {
    public int compareTo(T obj) { /* ... */ }
    // ...
}
```

Compilará el siguiente código? Si no compila, porqué?

```
Node<String> node = new Node<>();
Comparable<String> comp = node;
```

8.- Cuál será la salida del siguiente programa?

```
class Demo {
public void show()
   {
      ArrayList<String> list = new ArrayList<String>();
      ArrayList<Integer> list1 = new ArrayList<Integer>();
      boolean check = (list.getClass() == list1.getClass());
      System.out.println(check);
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

A. true
B. false