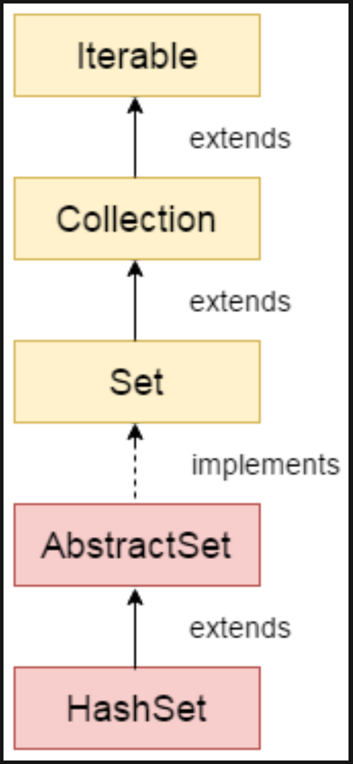
HASHSET

Es una colección de tipo conjunto que se utiliza para almacenar elementos

- No acepta duplicados
- No es ordenada
- Usa el equals y el hasCode para insertar
- No se usa para iterar
- Los objetos deben sobreescribir el hashCode para determinar duplicidades

HASHSET


Iterable
extends
Collection
extends
Set
implements
AbstractSet
extends
HashSet

HASHSET

```java
public class MainClients{
  public static void main(String args[]){
    Set clients = new HashSet();

    clients.add( new String("Rosita Alvires"));
    clients.add( new String("Porfirio Gonzales"));
    clients.add( new String("Agustin Jaime"));
    clients.add( "Agustin Jaime" );

  }
}
```

HASHSET

```java
class Client{
  String name;
  String lastName;

  Client(){ }

  Client(String name, String lastName){
    this.name = name;
    this.lastName = lastName;
  }
}
```

HASHSET

```java
public class MainClients{
  public static void main(String args[]){
    Client c1 = new Client("Rosita", "Alvires");
    Client c2 = new Client("Porfirio", "Gonzales");
    Client c3 = new Client("Agustin", "Jaime");
    Client c4 = new Client("Natalio", "Reyes");
    Client c5 = new Client("Natalio", "Reyes");

    Set clients = new HashSet();
    clients.add(c1);
    clients.add(c2);
    clients.add(c3);
    clients.add(c4);
    clients.add(c5);

  }
}
```

HASHSET

```java
public class MainClients{
  public static void main(String args[]){
    Client c1 = new Client("Rosita", "Alvires");
    Client c2 = new Client("Rosita", "Alvires");

    if (c1.equals(c2)){
      System.out.print("Son iguales");
    }else{
      System.out.print("Son diferentes");
    }

  }
}
```

HASH CODE

Es un número que hace referencia al objeto

HASHSET

```
class Client{
  String name;
  String lastName;

  public boolean equals(Object obj){
    if (obj instanceof Client){
      Client other = (Client)obj;

      if(this.name == other.name && this.lastName == other.lastName){
        return true;
      }else{
        return false;
      }

     }else {
       return false;
     }

   }
}
```

HASHSET

```
public class MainClients{
  public static void main(String args[]){
    Client c1 = new Client("Rosita", "Alvires");
    Client c2 = new Client("Rosita", "Alvires");

    if (c1.equals(c2)){
      System.out.print("Son iguales");
    }else{
      System.out.print("Son diferentes");
    }

  }
}
```

HASHSET

```
public class MainClients{
  public static void main(String args[]){
    Client c1 = new Client("Rosita", "Alvires");
    Client c2 = new Client("Rosita", "Alvires");

     System.out.print(c1.hashCode());
     System.out.print(c2.hashCode());

  }
}
```

HASHSET

```
public class MainClients{
  public static void main(String args[]){
    Client c1 = new Client("Rosita", "Alvires");
    Client c2 = new Client("Rosita", "Alvires");

    c1 = c2;

     System.out.print(c1.hashCode());
     System.out.print(c2.hashCode());

  }
}
```

HASHSET

```
HashSet()
HashSet(Collection c)
HashSet(int capacity)
```

HASHSET

```
void clear()
boolean contains(Object o)
boolean add(Object o)
boolean isEmpty()
boolean remove(Object o)
Object clone()
Iterator iterator()
int size()
```

EJERCICIO

Noughts and Crosses