https://leverageedu.com/blog/core-java-syllabus/

Java Editions

**The Java Programming Language has the following defined editions that it supports:**

Java SE (Java Standard Edition)

The [Java SE](https://www.javatpoint.com/java-se) is a computing-based platform and used for developing desktop or Window based applications. Thus, core Java is the part of Java SE where the developers develop desktop-based applications by using the basic concepts of Java where [JDK (Java Development Kit)](https://www.javatpoint.com/difference-between-jdk-jre-and-jvm#jdk) is a quite familiar Java SE implementation.

Java EE (Java Enterprise Edition)

Also known as **Java 2** Platform or [**J2EE**](https://www.javatpoint.com/java-ee). It is the enterprise platform where a developer develops applications on the servers, i.e., the enterprise development. This edition is used for web development.

Java ME (Java Micro Edition)

It is the micro edition that is used for the development of mobile phone applications. Thus, for the development of mobile applications, one needs to use [Java ME](https://www.javatpoint.com/java-me).

Thus, it is clear that Core Java is the part of Java SE and Java SE is the foundation for all other Java editions.

Concepts Covered in Core Java

**The following concepts are some of the major basic concepts of Java through which a beginner should go through:**

* Java Fundamentals
* [OOPs Concepts](https://www.javatpoint.com/java-oops-concepts)
* [Overloading](https://www.javatpoint.com/method-overloading-in-java) & [Overriding](https://www.javatpoint.com/method-overriding-in-java)
* [Inheritance](https://www.javatpoint.com/inheritance-in-java) with [Interface](https://www.javatpoint.com/interface-in-java) and [Abstract Class](https://www.javatpoint.com/abstract-class-in-java)
* [Exception Handling](https://www.javatpoint.com/exception-handling-in-java)
* [Packages](https://www.javatpoint.com/package)
* [Collections](https://www.javatpoint.com/collections-in-java)
* [Multithreading](https://www.javatpoint.com/multithreading-in-java)
* [Swings](https://www.javatpoint.com/java-swing)
* [Applets](https://www.javatpoint.com/java-applet)
* [JDBC](https://www.javatpoint.com/java-jdbc) (Basic Database Connections)

one can move towards the advanced Java version as the advanced section of the Java is quite interesting but can only be understood when the core concepts of Java are clear.

Core Java Vs. Advance Java

|  |  |
| --- | --- |
| **Core Java** | **Advance Java** |
| Core Java covers the basic concepts of the Java programming language. | Advance Java covers the advanced topics and concepts of the Java programming language. |
| Core Java is used for developing computing or desktop applications. | Advance Java is used for developing enterprise applications. |
| It is the first step, to begin with, Java. | It is the next step after completing the Core Java. |
| Core Java is based on single-tier architecture. | Advance Java is based on two-tier architecture. |
| It comes under Java SE. | It comes under Java EE or J2EE. |
| It covers core topics such as OOPs, inheritance, exception handling, etc. | It covers advanced topics such as JDBC, servlets, JSP, web services etc. |

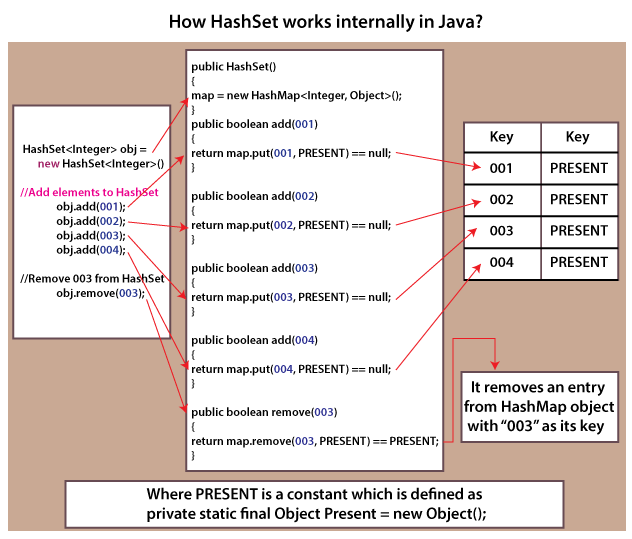
# Difference Between Set and Map in Java

In Java, **Set** and **Map** are two important interfaces available in the collection framework. Both Set and Map interfaces are used to store a collection of objects as a single unit. The main difference between **Set** and **Map** is that **Set** is unordered and contains different elements, whereas **Map** contains the data in the key-value pair.

In this section, we will discuss the Set and Map interface in detail also see the differences between them.

## Set Interface

The [**Java.util** package](https://www.javatpoint.com/java-util-package) provides the [Set](https://www.javatpoint.com/set-in-java) interface. Set is implemented by extending the collection interface. It doesn't allow us to add the same element to it. It doesn't maintain the insertion order because it contains elements in a sorted way. For designing the mathematical Set, we use the Set [interface in Java](https://www.javatpoint.com/interface-in-java).

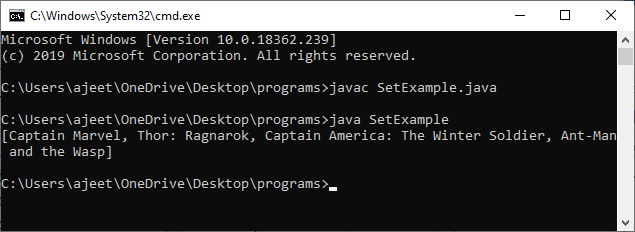


Let's implement the Set interface in a [Java](https://www.javatpoint.com/java-tutorial) program.

**SetExample.java**

1. **import** java.util.\*;
2. **public** **class** SetExample{
3. **public** **static** **void** main(String[] args)
4. {
5. // creating LinkedHashSet implementation using the Set
6. Set<String> marvel\_movies = **new** LinkedHashSet<String>();
8. marvel\_movies.add("Captain Marvel");
9. marvel\_movies.add("Thor: Ragnarok");
10. marvel\_movies.add("Captain America: The Winter Soldier");
11. marvel\_movies.add("Ant-Man and the Wasp");
13. System.out.println(marvel\_movies);
14. }
15. }

**Output:**

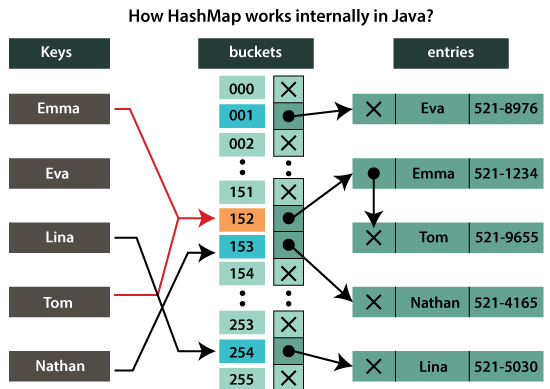


## Map Interface

[Map](https://www.javatpoint.com/java-map) is another important interface available in the Collection interface. So, in order to use the **Map** interface, we have to extend the Collection interface. Just like **Set, Map** is also used for storing collection of objects as a single unit. Each object is store in a key-value pair. We can easily access the value using just the key because each value is associated with a unique value.

We can easily search, update or delete the element by using the Map. **Map** and **SortedMap** are the two interfaces, and [**TreeMap**](https://www.javatpoint.com/java-treemap)**,**[**LinkedHashMap**](https://www.javatpoint.com/java-linkedhashmap), and [**HashMap**](https://www.javatpoint.com/java-hashmap) are three classes of Map.

If we need to traverse a Map, we have to convert it into Set because Map cannot be traversed. After converting it into Set, we can traverse it using the Set methods like **keySet()** and **entrySet()**.

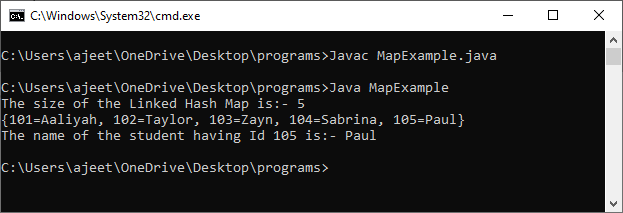


Let's understand how we can create a Map in Java by taking an example.

**MapExample.java**

1. **import** java.util.LinkedHashMap;
2. **public** **class** MapExample {
3. **public** **static** **void** main(String[] args)
4. {
5. // Creating an empty Linked Hash Map
6. LinkedHashMap<Integer, String> students = **new** LinkedHashMap<>();
7. // Adding data to Linked Hash Map in key-value pair
8. students.put(101, "Aaliyah");
9. students.put(102, "Taylor");
10. students.put(103, "Zayn");
11. students.put(104, "Sabrina");
12. students.put(105, "Paul");
13. // Showing size and data of the Linked Hash Map
14. System.out.println("The size of the Linked Hash Map is:- "+ students.size());
15. System.out.println(students);
16. // Checking whether a certaint key is available or not
17. **if** (students.containsKey(105)) {
18. String name = students.get(105);
19. System.out.println("The name of the student having Id 105 is:- " + name);
20. }
21. }
22. }

**Output:**



## Difference Between Set and Map Interfaces

Both interfaces are used to store the collection of objects as a single unit. The main difference between Set and Map is that Set contains only data elements, and the Map contains the data in the key-value pair, so Map contains key and its value.

Now, let's understand some major differences between both of them.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Set** | **Map** |
| 1. | Set is used to construct the mathematical Set in Java. | Map is used to do mapping in the database. |
| 2. | It cannot contain repeated values. | It can have the same value for different keys. |
| 3. | Set doesn't allow us to add the same elements in it. Each class that implements the Set interface contains only the unique value. | Map contains unique key and repeated values. In Map, one or more keys can have the same values, but two keys cannot be the same. |
| 4. | We can easily iterate the Set elements using the keyset() and the entryset() method of it. | Map elements cannot be iterated. We need to convert Map into Set for iterating the elements. |
| 5. | Insertion order is not maintained by the Set interface. However, some of its classes, like LinkedHashSet, maintains the insertion order. | The insertion order is also not maintained by the Map. However, some of the Map classes like TreeMap and LinkedHashMap does the same. |