**PermGen**

* (Permanent Generation) is a special heap space separated from the main memory.
* The JVM keeps track of class metadata in the PermGen. Also, the JVM stores all the static content in this.
* Due to limited memory size, PermGen can throw OutOfMemoryError.

**Metaspace**

* Metaspace is a new memory space.
* It has replaced the older PermGen memory space.
* It can now handle memory allocation.
* Metaspace grows automatically by default.

MetaSpace:

Stores general metadata.

Static variables are stored.

Permgen:

It is replaced by metaspace from java 8

Functional Interface:

To achieve 100% abstraction.

|  |
| --- |
| **Till java 7:** Only one abstract method will be there No Concrete methods allowed.  **From Java 8:** Concrete methods are allowed in the form of default methods |

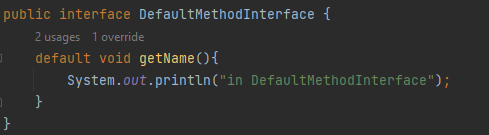
Public static (for methods)

Public static final (variable)

Default methods in Functional Interface:

There will be no restriction to override Default methods in the child classes.

You can use default behaviour or you can override and specify your own implementation.



A screen shot of a computer

Description automatically generated

Output: in DefaultMethodInterface

A screen shot of a computer program

Description automatically generated

Output: in ClassOverridingDefaultMethod

Default Methods in Multiple Inheritance:

If you have method with same signature in 2 different interfaces and extending them at same class will cause Ambiguity issue.

To solve this we have 2 ways:

* 1. Provide your own implementation in the child class.
  2. Use InterfaceName whose implementation you want along with super and method name.

ILeft.super.m1()

A screen shot of a computer code

Description automatically generated

Static Methods inside Interface:

General utility methods we can use in that form of static methods.

Static methods in interface can only be accessed by using Interface name.

static methods are not accessible to child classes so there is no point of Overriding Concept.

A computer screen shot of a code

Description automatically generated

**Method Reference & Constructor Reference (::)**

Alternative syntax to lambda expression.

Instead of providing new implementation, I can reuse already existing implementation.

Code Reusability.

**Method Reference:**

**Note**: Both methods should have same argument types. (return type, modifiers etc need not to be same)

**Static Method**:

ClassName::methodName

Non-Static Method:

ObjectRef::methodName

**Constructor Reference:**

Collection & Stream

Collection:

Groups of objects as a single entity.

Steams:

To process the objects in the collection.

You can use for arrays as well by using stream.of()

To perform bulk operations.

Filter:

Based on condition perform operation.

It used Predicate(takes input and return Boolean value)

Map:

Transform each item in the collection.

It uses Function(takes input and returns output)

collect():

Used to collect the returned items from stream.

count():

sorted():

1. sorted() -> natural sorting
2. sorted(Comparator c) 🡪 for customized sorting order

min():

min(Comparator)

max():

max(Comparator)

forEach():

use Predicate.

toArray():

Date&Time API