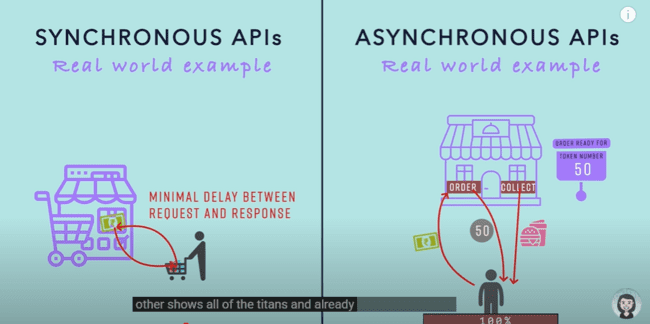
**Asynchronous and synchronous API calls:**

When you purchase an item from a grocery store, then it would be syn call, as you get the things when you order and pay.

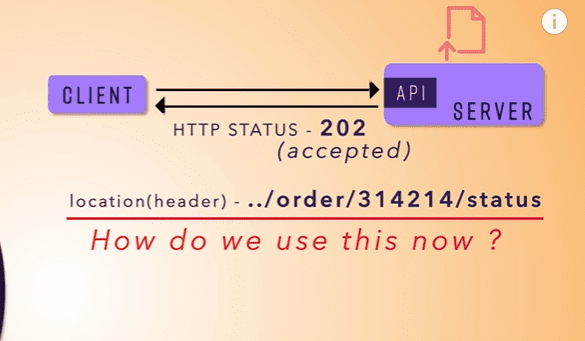
But with Async, when you go to the restaurant, where self service is in place, there you get the token or bill when you pay, and you need to wait for the food to prepare and then you’ll get the food.



In case of a asynchronous API call, where we need to upload a large file or data, say 1GB, then we need to send out the response as 202(Accepted), instead of 200(OK).

Now in this case, we would be given back a token in the http header along with a status code.

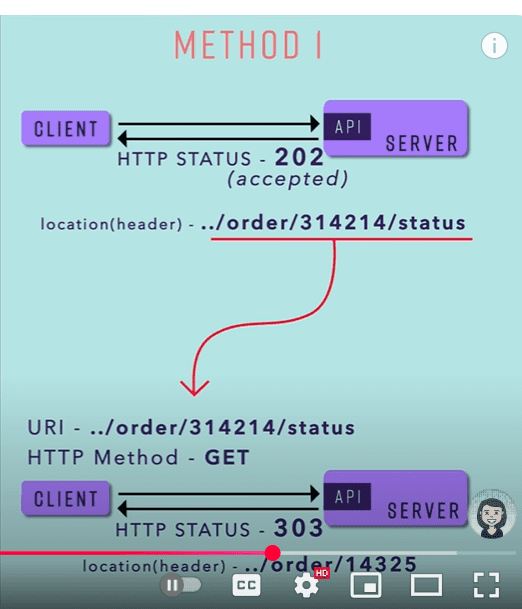
And with that token, we can request the status of the uploading of that file, what is the status of that which would either Failed, Completed or In progress.



A screenshot of a computer

AI-generated content may be incorrect.

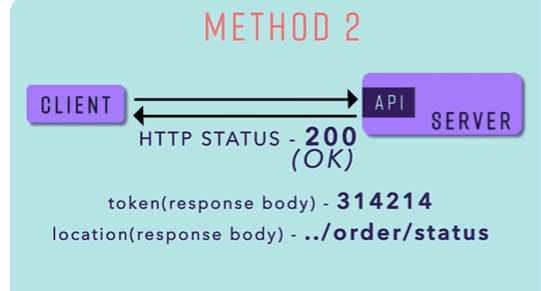
If the status is completed, then there might be in http status code 303, which tells that operation has been completed and resource has been uploaded on a specific location, which has been shared in the httpheader response of the API call.



So this is the one method to send out the response for uploading the data.

There is another way where we don’t need to send out the data in the response header and no need to send out the 303-response code.

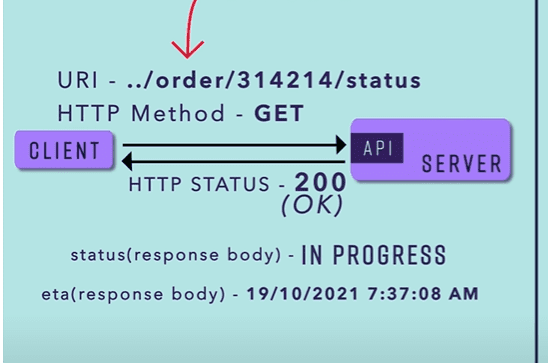
We can send out this data in the response body like following:



We get the data in the response body, as the link of the uploading data, where we can find out the status of the call.

When we try to access that link, we can similarly send out the response in the response body as following with different status:

ETA with In progress status:



Completed status with file location status:

A diagram of a method

AI-generated content may be incorrect.

Failed status code with reason:

A diagram of a method

AI-generated content may be incorrect.

Use cases:

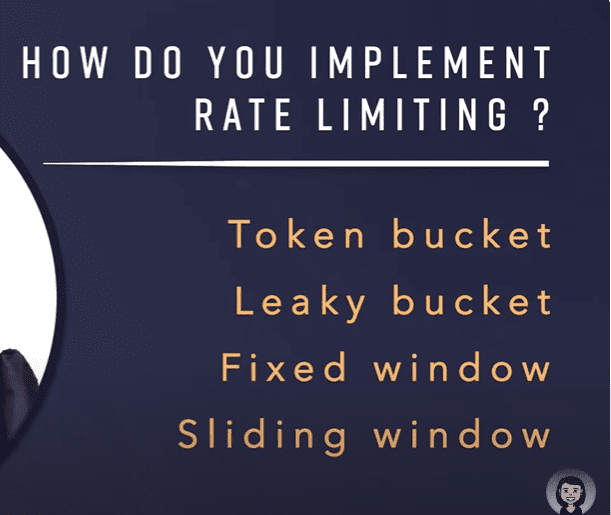
1. When the response time would be high or there might be latency in case of the response, then we need to use the asynchronous API call.

**API RATE LIMITING**

Limiting the access like can only hit 10 calls per day for the API.

429 is the error we use for the Rate limiting or for the too many requests.

Following are the ways to implementing rate limiting:



There are two types of pagination:

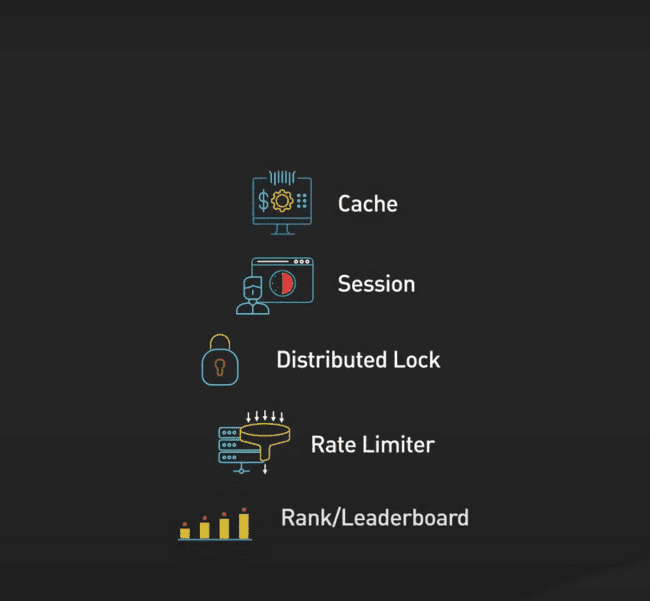
1. Page + offset
2. Cursor based.

HTTP server by default handles 200 requests.

And 8 requests, which our system can handle.

TOP USE CASES OF REDIS ARE:

REDIS cluster and sharding.



Tools for multiple request testing:

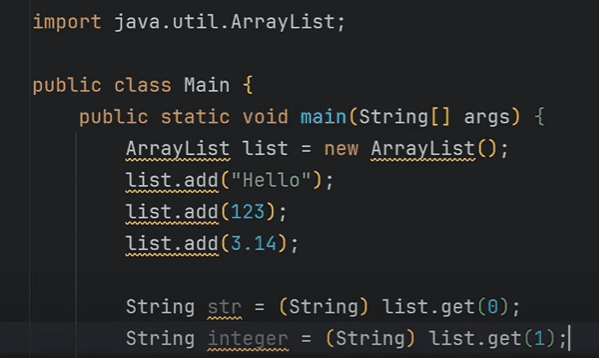
1. HEY
2. Apache BenchMark
3. Jmeter

Java virtual threads in spring Boot

MESH SERVICES in Microservices.

**Class is the blueprint for the object, in the similar way, interface is the blueprint of the class.**

TYPE SAFETY AND MANUAL CASTING is the issue when we use the List with Object type as:



So we need to use the diamond operator, to describe the arraylist<>.

A black background with white text

AI-generated content may be incorrect.

Now if we have Box class like this:

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Now if give this a String, it would provide the class cast exception, as we are setting up the value as 1.

A screen shot of a computer program

AI-generated content may be incorrect.

So, we need to use the GENERICS and convert the same class to the generics type as shown below:

A screen shot of a computer program

AI-generated content may be incorrect.

Now, it becomes type safe, and we need to explicitly provide the data type, which needs to be given while initializing the class.

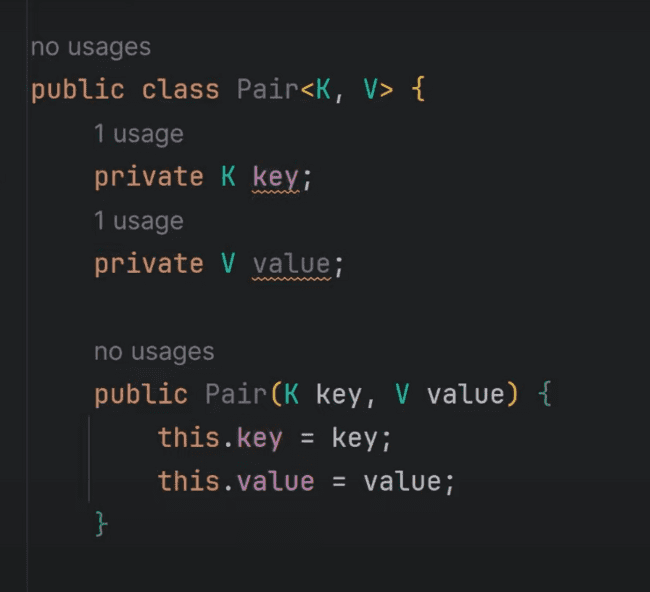
A screen shot of a computer program

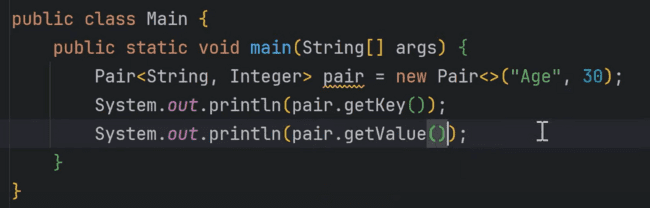
AI-generated content may be incorrect.

In place of <T>, we can also replace this with the <P> ,<PL> or anything, but the convention is, we need to write the <T>.

And the advantage is that we can make the Box object with String, with Integer, or with any type we wanted.

And for the two types of datatypes we can use the class as following:





These are conventions, we need to use when we need to use the generics.

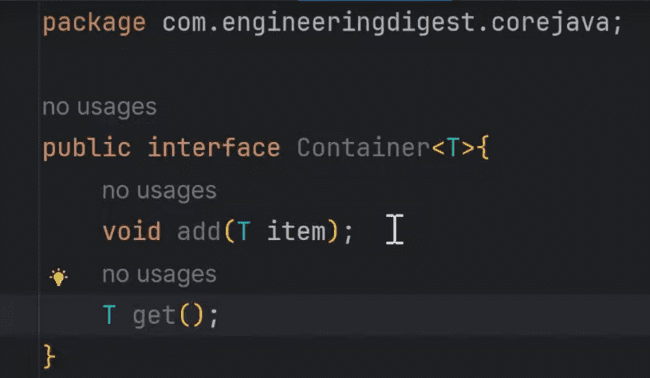
A screenshot of a computer

AI-generated content may be incorrect.

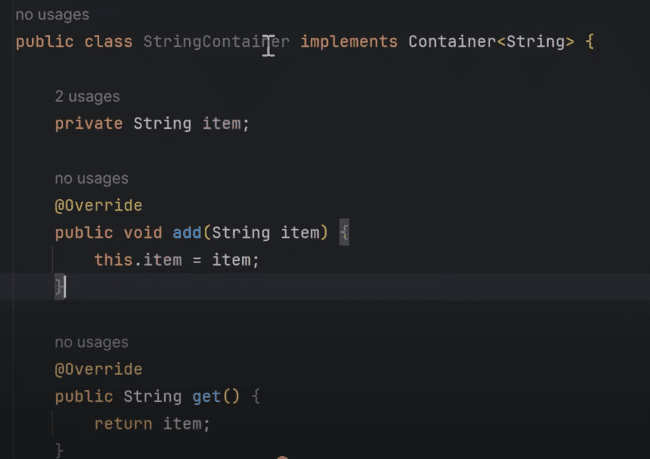
T for the single class, <K,V> for the key, value pairs, E for the collection types.

**GENERICS INTERFACE:**

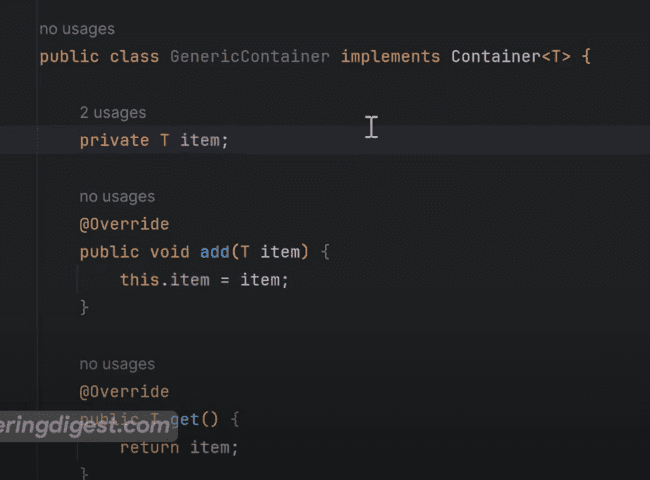
Example of the interface:



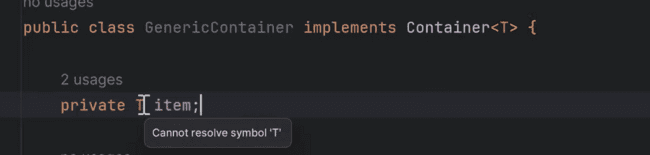
We can make the StringContainer using the above interface, where we need to provide the implementation of the StringContainer as shown below:



Or we can make that as the GenericContainer for the generic interface as shown below:



IMP: But there is one error in the above implementation that, T symbol is not resolvable:



So, we need to provide the <T> with the GenericContainer as well as shown below:

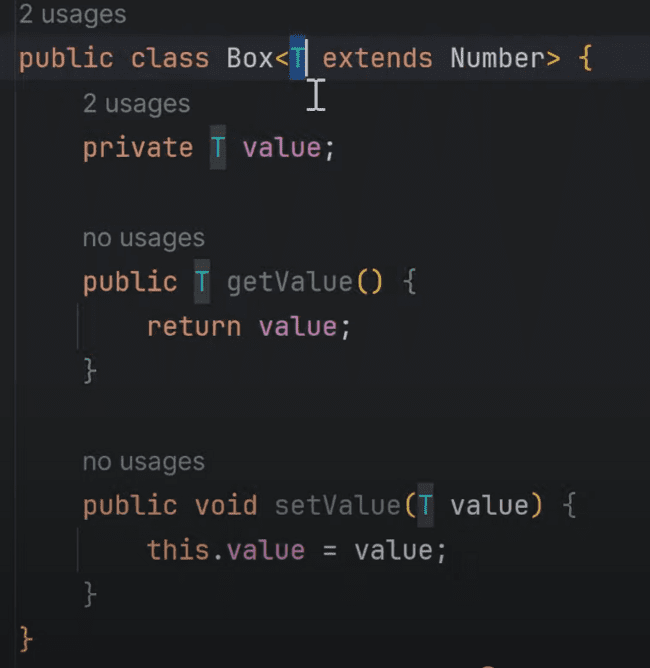
A screen shot of a computer

AI-generated content may be incorrect.

**BOUNDED TYPE PARAMETERS:**

Here we can bounded the generic with some of the conditions, that whether it needs to extend some class or it can implement a particular interface.

For the below case, we have bounded the class to extended from Integer, Long, Double and other data types, but with String, Enums:



With Integer it is valid, but not with String.

A computer screen with text

AI-generated content may be incorrect.

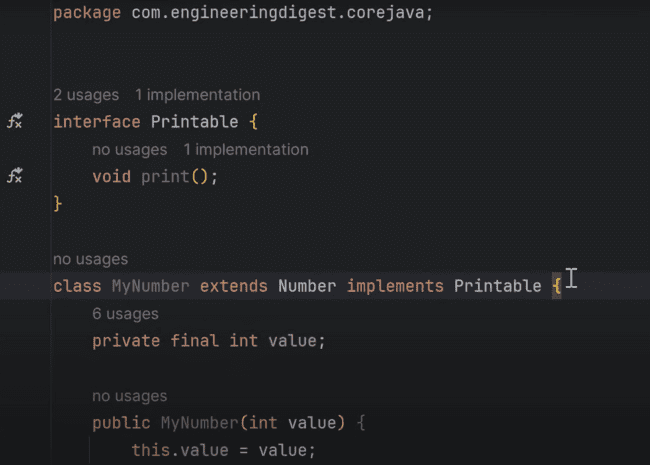
A computer screen with text and symbols

AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

We can also provide the multiple constraints on the <T> as well.



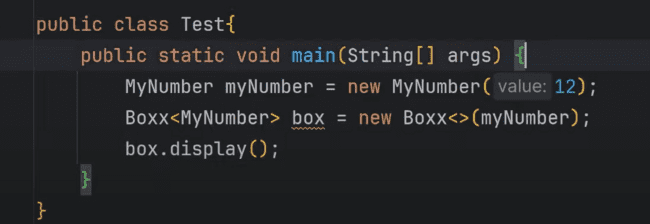
So, we can make a class which can write like this one.

A black background with white text

AI-generated content may be incorrect.

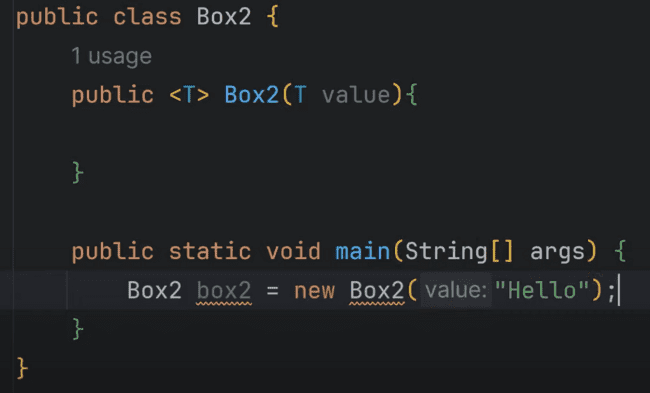
And, also we need to provide the interface after the & symbol, this is due to the reason that, we can implement multiple interfaces in Java, but class we can extend only one.

We can make the object like this one.



**GENERIC CONSTRUCTORS:**

Now if we want that the class needs not to be generic, but its constructor can be generic, then we can write like the following:



In the similar way, integer can also be passed in the constructor as shown:

A computer screen shot of text

AI-generated content may be incorrect.

We can also provide the bounded arguments as well in the constructor.

A screen shot of a computer program

AI-generated content may be incorrect.

**GENERIC METHODS:**

Method signature of the generic method which is printing the array of any type:

A black screen with white text

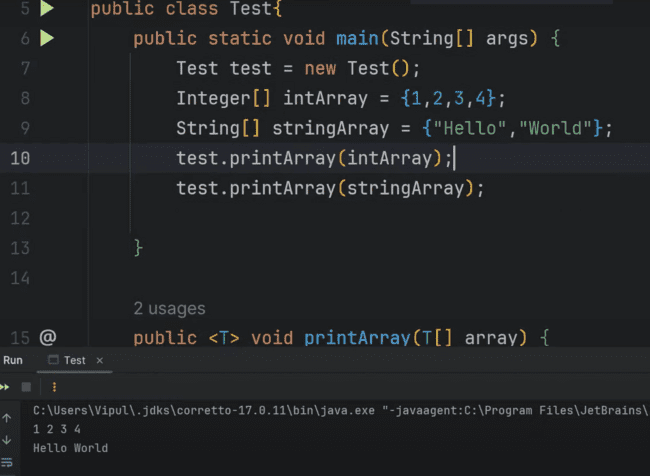
AI-generated content may be incorrect.

T[] is the type array and also we have T element for any kind of the element as shown below:

A screen shot of a computer code

AI-generated content may be incorrect.

Passing the string and the integer array in the main class as:

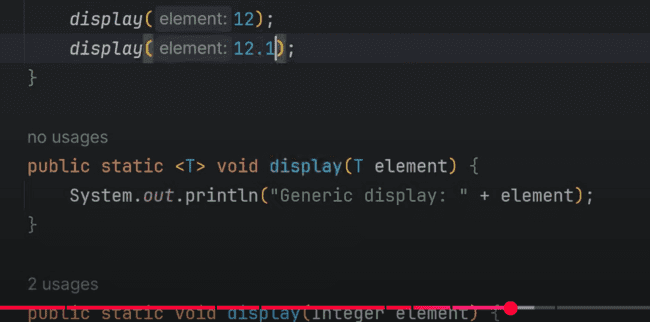


Method overloading in the generics:

A screen shot of a computer

AI-generated content may be incorrect.

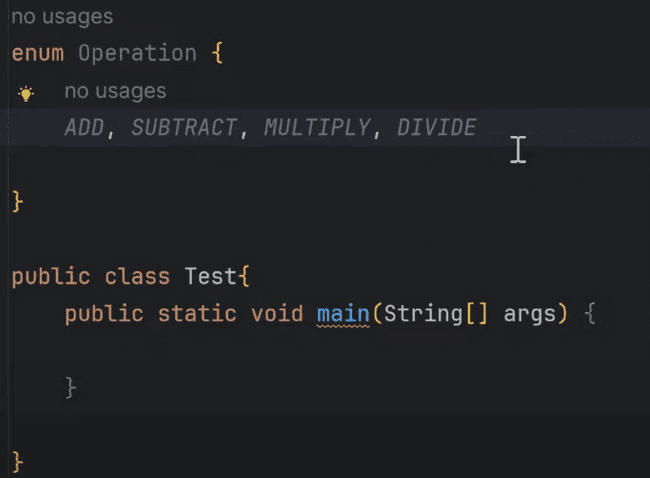
If we pass the integer, then second display method will be called, while if we called the decimal number or String, then second display method will be called.



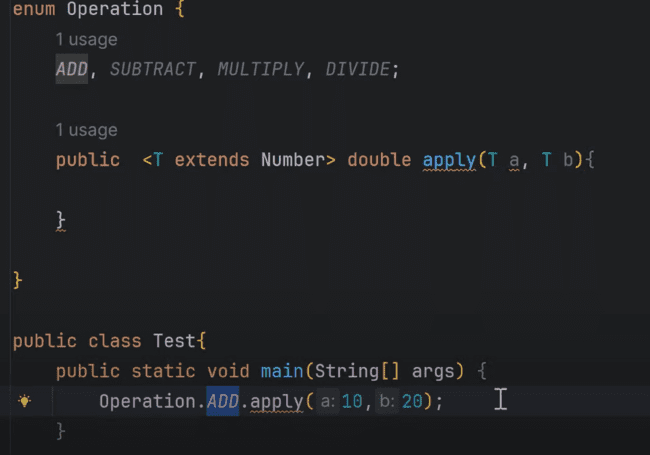
**GENERICS IN ENUMS:**

**A computer screen with text and images

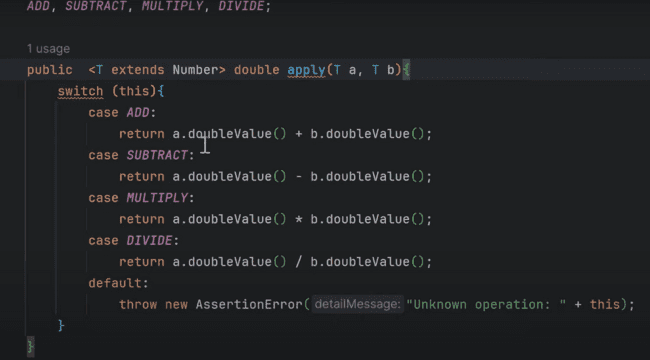
AI-generated content may be incorrect.**

****

For the above screenshot, if we want to provide the implementation for the add, subtract, multiple and divide in the Enums, then we need to use the generics, otherwise, we can use different methods for the same as shown below:



Simply we can use this, and provide the implementation using the switch statement as shown below:

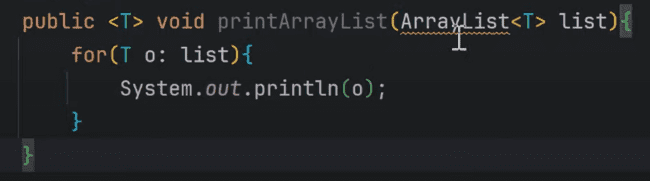


**WILDCARDS IN GENERICS:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

In the following case, we are not returning anything, we are only printing the arraylist, so we can use the wildcards:



So, we can use the wildcards in place of that, and use the following:

A computer screen shot of a program code

AI-generated content may be incorrect.

**IMPORTANT: As, wildcards are used only for read only purpose, If we try to use the wildcards pattern, when we are trying to add the two things, then we can not use the wildcards pattern as shown below:**



We get the issue when we use wildcards here:

A screenshot of a computer program

AI-generated content may be incorrect.

We can define the list using the wildcards:

A computer screen shot of text

AI-generated content may be incorrect.

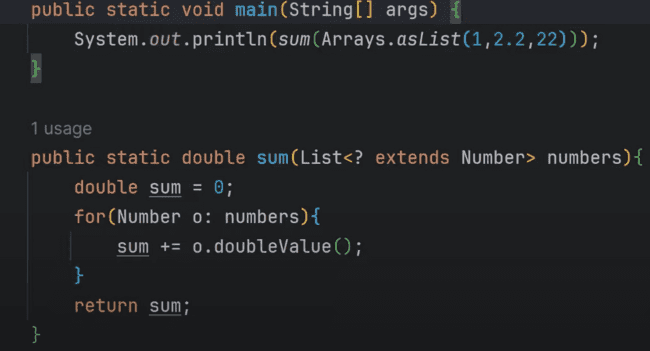
But when we try to add the objects using the list, then we’ll be getting the issue as shown:

A computer screen shot of text

AI-generated content may be incorrect.

***UPPER BOUND:***

We can use the upper bound with using the keyword extends, when the wildcard(?) can be the subclass of the defined superclass NUMBER and it can be Integer, Double, Long and others as well.



***LOWER BOUND:***

In the Lower bound, we can provide the provide the super class of the defined class in the wildcard means floor as shown below:

A computer screen shot of a black background

AI-generated content may be incorrect.

Now even if are unable to add using the wildcards, but there is one condition still where we can be able to add the things, when we provide the lower bound and upper bound while declaring the list as shown in the below screenshot:

A computer screen with text

AI-generated content may be incorrect.

***TYPE ERASURE: HOW GENERICS WORKS INTERNALLY***

Here we would be seeing how generics works internally. In this case, the JVM internally updates the GENERICS like BOX<T> with BOX, and cast the object when the object is getting created, but this happens internally, when the byte code is getting generated.

***GENERIC EXCEPTIONS:***

Generic exceptions: Due to type erasure, and since exceptions are closely tied to the runtime operations, so generics won’t work with exceptions.

A black and white text

AI-generated content may be incorrect.

So, we can create custom exceptions and use the generics in their constructor to handle the exception with any kind of type like:

A screen shot of a computer code

AI-generated content may be incorrect.

Now we can call this exception with either String or Integer and throw them explicitly as:

A screen shot of a computer program

AI-generated content may be incorrect.

After running the class, the output would be:

A screen shot of a computer program

AI-generated content may be incorrect.

***ENUMS IN JAVA:***

Enumeration means listing things. Like months in a year, departments in college, days in a week.