Interface:-

What we have in java 7? What are new features of java 8?

Java 7 :- we are having only public methods which are abstract by default

We are using anoymus class

With help of abstract class or interface we create anonmus classes

Features java 8

In java we are having lambda expression

We also having default methods in interface when we work in java 8

We also having static methods in interface when use java 8

With default method we can declare a method and provides it body in interface and we can call them like any other method of the class

A class which implements the interface can access the defaults without overriding them.

It means we don’t need to override the default methods

When we talk about static methods we only learnt that static method can be part of class but in java 8 they made a change with that we can declare a static method inside any interface and we can call them using interface name as we call any static method of class.

Interface SI1

{

Static void show()

{

Code to be perform

}

}

My question

We are having static and default methods in interface and same in class so what is the difference between interface and class after java 8.

Ans: we can not declare a constructor in interface but we are having constructor in class always.

We can declare constructor inside the abstract class.

When we talk about abstract class we just that we can not create its object.

We can not non abstract methods inside the class and we can declare the variables inside the class too.

Functional Interface:-

A functional interface will have only one method in it. With functional interface we can implement lambda expression.

With the help of lambda expression the problem of multiple annonymus class is solved.

Interface f1

{

Void test();

}

F1 Is functional interface as it Is having only one method

Syntax of lambda expression :- ([argument-list])->{ body }

1

1. Argument list is optional as if declare arguments inside function then we can use them in lambda expression
2. Arrow -token it Is use to link the argument list and body of expression
3. Body it actually contain the code of function which is going to be called.

With functional we can perform inheritance but in such cases we the second interface will not have any other methods.

Lambda expression say we can use it with the help of functional interface. And functional interface can have only one abstract method.

Interface l2

{

Void show(datatype var);

}

We are having forEach();

First of all we will create a list of student name

Then we will call list.forEach()

Inside forEach we will call lambda expression to print the values of list

forEach(nameoflist:var);

forEach(Iterable interface and we declare the method of iterable interface in lambda expression)

forEAch(Classname <super >:action)

forEach( (n)->System.out.println(n) );

for( datatype var : datatype Listobject)

we can also creates the threads using lambda expression

the process is

we are having a interface name Runnable

object of Runnable interface is passed to Thread class to create the object and start the thread.

Class MyThread implements Runnable

{

Public void run()

{int i;

While(1)

{

For(i=1;I;i++)

System.out.pritnln(i);

}

}

Public static void main(String s[])

{

MyThread t1= new MyThread();

Thread tobj= new Thread(t1);

Tobj.start();

}

}

Runnable r1= ()->{

System.out.pritnln(“Lambda expression is used to create the thread”);

};

Thread t1= new Thread(r1);

T1.start();

This is for lambda expression and functional interface.

Date :- java 8 we are having stream api

What is stream api?

It is same as the pipe line it provides different output as per our need

In general pipes are may provide output in different manner if put a shower at the outlet of pipe the water output is different in style.

Arraylist , vector so in that case we can go for stream api to provide.

Why we use stream api.

As I said with arraylist we need to provide iterator but if I want data should be filtered as per our condition. So what we need to do we will write some of the if else condition then we can see the data.

Filtering

One more thing stream does not store any data, it is lazy , it will evaluates the code only when it is required.

Stream we can find inside java.util.stream

We can perform following task from stream we can filter, collect , print or we can convert the data from one data structure to another

Java stream interface with following methods

Stream.Builder builder ()

Collect (Collector <? Super T,A,R> collector)

long count()

distinct () it will work same as we are doing with select distinct in sql command

filter

findAny()

findFirst()

flatMap()

DoubleStream flatMapToDouble

forEach()

forEachOrdered()

iterate

max

min

sorted

toArray()

toArray()

reduce(identity, function super, binaryoperator)

Assignment :

1)Use filter method to print even no out of a arraylist

2)use filter method to save object of Employee class, on following condition those employee having salary > 15000. Print the list of all employee after applying stream.

3) use filter method to sum all employee whose salary <15000.

4) use filter method to print name of all employee in capital letter

**Time date api java 8**

Java.time, java.util and java.sql which contain clasess for date and time

Java 8 we also get new date/time API

Java.lang.system which provide currentTimeMillis() after 1st January 1970

Java.util.Date specific instance of time, with unit millisecond

Java.util.Calendar which is an abstract class

Java.text.simpleDateFormate

Java.util.TimeZone it provides different time zone values as we all knows whole world is divided in different timezone

Current time in Bharat is 4:26 PM

China around 45 min ahead of us

Dubai is around 90 min backof us

UK is around 5:30 min back from us means that when we are having 5:30 Pm in Bharat. In UK time is 12 Noon

**There are not Thread safe**

**Api not design**

**It is difficult to handle the time zone.**

**Java 8 Date Time clear all the problems**

**Java.time.LocalDate**

**localTime**

**LocalDateTime**

**MonthDay**

**Clock**

**ZonedDateTime**

**Year**

**YearMonth**

**Duration**

**Java.time.DayOfWeek enum**

**Java.time.Month enum**

**Java.time.LocalDate :- atTime(int hour, int min) it is use to combine date and time**

**Int compareTo(dateobject )**

**Equals(object)**

**String Format()**

**isLeapYear()**

**minusDays(long days)**

**of(year,month,day);**

**LocalDate obj2= obj.minusDays(20);**

**Obj =”28-1-22”;**

**Obj2 will contain 20-1-22**

**plusDays**

**it will give us the date after adding no of days to current date object**

**ZonedDateTime**

**format() we can format our date time**

**get() it will us value of specific field from current date time**

**getZone()**

**we are having of method from LocalDate same method is here which provides us facility to obtain instance of zonedDateTime from local date time**

**minus**

**plus**

**Date 31-1-22**

1. Write down code which will take input of names in Arraylist and check a each word either it is palindrome or not.
2. Write down code to find he maximum out of list using lambda expression
3. Take date object from java.time api and store them and sort them and apply following operation. Sort date, print them. Find a date out of list.

1-2-22

Spring :-

Dependency in our project or in our programming

As all of us know how we book ticket of train or airlines prior to internet application

What we do we need to visit the booking counter after that we can only book the ticket. In booking a ticket through counter we are tightly coupled with the booking counter.

Later we got the websites by which we can book the tickets but the problem is same that we need a computer and internet connection

But today as we all know that we are having smart phones by which we can book the ticket at any time from anywhere. So now we are loosly coupled with booking counter

Class A

{

B obj= new B();

Void show()

{

Obj.display();

}

}

Class B

{

B(int a,int b, float f)

{

}

}

Class A

{

B obj=new B(12,22,23.4567);

Void show()

{

}

}

How we will remove tight coupling? See the code

Class A

{

B b;

A(B objB)

{

B=objB;

}

}

Class B

{

B(int a,float f, char c) { }

}

public static void main(String s[])

{

B b = new B(12,34.433,’a’);

A obja= new A(b);

}

We discuss how we are going to use this loose coupling in our programming

Some design patterns :- IOC

Inversion of control it means we just go for dependency injection

Spring framework :

By which we can perform DI , IOC,

We got so many features of spring

Spring is having predefined Templates :- hibernate, jpa, jdbc, etc technologies.

Loose Coupling:- spring applications are loosely coupled .

It is very easy to test the spring application

We can say that spring application is light weight. We are having POJO implementation in our spring application. Spring does not required so many classes to be inherit. When a application does not inherits to many classes it is said to be lightweight.

We can develop application a very fast speed.

Strong abstraction for JMS, JPA, JDBC, JTA, etc.

We can perform caching, validation, transactions and formatting for declarative support .

If we want to combine the spring with struts application we can also go for that.

Spring module

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | We are having data acess or integration  Jdbc, orm or jms, transactions |  | Web (MVC/Remoting)  Web, servlet  Portlet, struts |  |  | | --- | | Aop, aspects, instrumentation |  |  | | --- | | Spring core container  Core, beans, context, expression language | |
| Test case |

Spring core :- it can be developed in any editor like sts, eclipse, or netbeans, intelij

We are going to learn how we inject dependency through setter and getter methods,

And one more way is constrctor.

So we can execute our prog using these two methods

We need one file say it is class A, Class B.

Class A is dependent on Class B.

We will pass the object of class B to class A. either by setter method or constructor.

We have to load the object of class A. it means we are not going to create object

We are using spring core jar files which helps us in loading the objects through beans.xml file or we can say that it is our applicationcontext file

We need

Class A, class B, one class which is having main method and another one is applicationcontext.xml file

So what will be the flow of our application

We are going to write the dependency inside class A and we will just pass the values to class A through applicaitoncontext.xml file and it will be loading object for us

So in our first example we are just setting the variables of any class

IOC Container It is responsible for executing our codes in spring loading beans, putting dependencies and providing support for all kind of injections

Two container

BeansFactory

Applicationcontext: - it Is better to use application context as it is built on beansFactory so it provides much more functions than bean factory.

Inside our applicationcontext.xml file we are going to create the

<bean id=”using this id we will load the bean class in our main or in our application” class=”package.beanclassname”>

<property name=”” value=””></property> :- this will be used when we are passing values by setter methods

</bean>

One of the way to set the variables is using setter method which we have seen

Second one is constructor method

In constructor method we need to write as many constructor as many variables we are having

We have just seen how to set variables of a class

Now we will go to set or inject dependent object

We will create a class name

Address

We will inject address class object to our employee class

If I am a variable name id

Setter method will be setId()

If a variable name is addresss

setAddress

now wewill learn how to use the collection framework inside applicationcontext.

Or we can set collection object using dependency injection

I will create a class Name person, person will have name, hobbies

Autowire

When we use autowireing feature of spring framework it is very easy for us to inject the dependent object in our class.

Interally it is uses setter or constructor.

But autowiring we are not able to inject the primitive values to our class we can only pass the object so it is very easy in terms when we talk about jdbc or jpa or hibernate.

5 type of autowiring

1. NO :- no autowiring
2. byName:- it means dependent object is matched with name
3. byType :- in this type of object is searched and injected
4. byConstructor :- in this type we inject object using constructor mode
5. autodetect :- it is deprecated since spring 3.

Steps:- first create the class which is going to be injected

Create second in which object to be inject

Now mention the name of bean in applicaitoncontext.xml file

Load the bean from xml file

No unique bean definitionException

Why autowiring is required?

We are having sql connection which we need to connect and configure each time we change our server or env.

With autowire mode byType it is not required to have injected bean have same name as in applicationContext file