Section 3 continuation

271. When should I use While loop?

Project (corejava) Class ( whiledemo)

So the basic syntax of while loop is while of Boolean expression, here Boolean expression, if it returns true then only the control will go inside this loop, if the expression returns a false then the control will not go inside the loop only this condition is satisfied, then only it goes inside the loop

I want to get the, printing the numbers from 1 to 10 sequentially, how would I print that using while loop?

First  I would create a variable called i and I will initialize it to zero ( int i=0; )

Int i=0;

While(i<10)

{

System.out.println(i)

I++;

}

when I run this program it'll goes to infinite loop by printing zero. Bcze 0<10 true.

until unless the condition becomes false in this argument, this loop will never stop executing.

Now, I would increment after printing the i variable ( i++; ) i plus plus in the sense,

zero will increment by one value and now i becomes 1 here.

Int i=10; (i>0) ( i--)

If you want to make it less than zero, decrement rather than incrementing and now when you say i--, it'll keep on decreasing and after completing one, 10 will become 9, 8, like that ( I--; )

272. How do while loop works?

I want to print the numbers from 20 to 30.

Syntax of do while loop

Int j=20;

do

{

System.out.println(j);

J++;

}while(j<30); // one loop of execution is guarantee in do-while loop

Once it reaches 29, once after that it reaches 30, so 30 less than 30, loop will stop executing.

The basic difference between while and do-while is

So in while it will not execute without evaluating the Boolean expression,

but whereas in do-while, first it executes for one loop, and then it evaluates to run for more loops. So that means one loop of execution is guarantee in do-while loop.

Because without checking any Boolean expression, first it is running a block of code.

And now for next iterate, it's comparing and then proceeding for next iteration.

So one, it is allowing, but whereas in while loop, you cannot run the block without evaluating.

274. Explanation on Nested for loops ( very important )

what if I write one more for loop inside this block?

for(int i=1;i<=4;i++) //(outer for loop) loop will run four times

{

System.out.println(“outer loop started”);

for(int j=1;j<=4;j++) //inner loop

{

System.out.println(“Inner loop”);

}

System.out.println(“outer loop finished”);

}

First to finish one outer loop, I have to complete all four inner loops.

four times when I run the outer block, each time I have to run this inner loop also.

That means four into four, 16 times I have to run this inner the loop to just finish four times of this outer loop.

If you want to debug your script, that means if you want to go step-by-step how Java is moving into your script. Just double click on any of the line breakpoint

// 1 to 10  
  
/\* for(initization;condition;increment)  
{  
  
}\*/  
/\* if(5>2)  
{  
System.out.println("success");  
System.out.println("second step");  
}  
  
else  
System.out.println("fail");\*/  
  
  
for(int i=0;i<10;i=i+3)  
{  
if(i==9)  
System.out.println(" 9 is displayed");  
else  
System.out.println("I didnot find");

275. Practise Exercise on loops -1 { printing pyramid triangle)

When outer loop is incrementing by one number, we should decrement one inner loop of running.

system.out.println in the sense every time ( we are getting all these in separate lines )

“ln” stands for print it in the new line. If you remove “ln” then then it will come line side by side.

“\t” - will give you the spaces

//nested loops works -  
int k=1;  
for(int i=0;i<4;i++) //(outer for loop) this block will loop for 4 times  
{  
//System.out.println( "outer loop started");  
for(int j=1;j<=4-i;j++) // inner loop   
{  
System.out.print( k);  
System.out.print("\t");  
k++;  
}  
//System.out.println( "");  
}  
}

Output:

1 2 3 4

5 6 7

8 9

10

276. Practise Exercise - 2- Yahoo Ques : ( Inverted sequence pyramid logic program)

int k=1;  
for(int i=1;i<5;i++)  
{  
for(int j=1;j<=i;j++)  
{  
System.out.print(k);  
System.out.print("\t"); // will give you the spaces  
k++;  
}  
System.out.println( "");  
  
}

Output:

1

2 3

4 5 6

7 8 9 10

//Int k=1;

for(int i=1;i<5;i++)

{

For (int j=1;j<=I;j++)

{

System.out.print(j);

System.out.print(“\t”);

//k++;

}

System.out.println( “”);

}

Output:

1

1. 2

1 2 3

1 2 3 4

278. What is interface?

It's also one of the core concept in Java OOPs, in object oriented programming.

So interface is, it lies on the same lines of class. It's not big difference between these two,

but the only difference between interface and class is interface will have methods, but not a body.

So in class, we generally define a method and we start writing code into it.

Ex: every country have their own traffic rules, So, but there is one common centralized traiffic rule which should be followed by all the countries in the world. ( green-go,red-stop,flashyellow wait)

Top over these, you can add your own rules with your country specific, but there is some base rule which should be followed by all the countries across the world.

Austraila - have there own rules to define, but I have to make sure that I am following these three common rules imposed by central team.

For that reason, central team, they will define interface.

And say they write these three methods inside it and they'll ask us to implement these methods in your class. so it is an agreement

client interface and class to implement the methods present in this interface.

So classes should define and implement methods present in interface.

Basically the classes should write body of interface.

Only classes can implement the methods present in interface, that you should remember.

Now as in country specific traffic, first of all I need to implement all the methods present

in that central traffic.

So to do that, to implement an interface in your own class, you can do that by writing a keyword called implements and then write that interface name( centralTraffic ).

If u not added the implemented methods in class it will throw error like “Add unimplemented methods”

279. How java classes can take advantage of interfaces

if you want to execute these methods you need to create a class object for it.

I have created object for this class, but here this object is referring to the methods in Australian traffic. But we are actually referring to the methods present in central traffic,

In that case, if you are implementing an interface and you are writing the methods related

to that interface method, you could just say here central traffic rather than Australian traffic.

And these interfaces can also take variables. We can also define variables also here,

but you have to make sure that these variables are public. ( int a=4;//public )

So if you don't define them as a public even if you don't use that,

this will treat these variables as public,

Are variables can be defined interface? - Yes. If so, what is that access modifier?

The public is the access modifier.

But just remember that all the variables defined in this interface are public in nature.

After implementing the methods present in interface, you could define your own methods

which is specific to your Australian rules.

one class can implement multiple interfaces

280. Interface code download

package coreJava;

import demopack.CentralTraffic;

public class AustralianTrafic implements CentralTraffic,ContinentalTraffic {

public static void main(String[] args) {  
// if you are implementing an interface and you are writing the methods related

to that interface method, you could just say here central traffic rather than Australian traffic.  
CentralTraffic a= new AustralianTrafic();  
a.redStop();  
a.FlashYellow();  
a.greenGo();  
  
AustralianTrafic at=new AustralianTrafic();  
ContinentalTraffic ct=new AustralianTrafic();  
at.walkonsymbol();  
ct.Trainsymbol();  
  
  
}

@Override  
public void redStop() {  
// TODO Auto-generated method stub  
System.out.println(" redstop implementation");  
}  
public void walkonsymbol()  
{  
System.out.println("walking");  
}  
@Override  
public void FlashYellow() {  
// TODO Auto-generated method stub  
System.out.println(" flash yellow implementation");  
//code  
}

@Override  
public void greenGo() {  
// TODO Auto-generated method stub  
System.out.println(" Green go implementation");  
}

@Override  
public void Trainsymbol() {  
// TODO Auto-generated method stub  
  
}

}

//CentralTraffic

package demopack;

public interface CentralTraffic {  
int a=4;//public  
public void greenGo();  
public void redStop();  
public void FlashYellow();

}

//ContinentalTraffic

package coreJava;

public interface ContinentalTraffic {

public void Trainsymbol();  
}

281. What are Abstract classes and how different they are from Interfaces

What is Abstraction ?

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

Abstract Classes & Methods :

Class which is declared with the abstract keyword is known as an abstract class in java.

It can have abstract (method without the body ) and non-abstarct methods ( method with the body )

Abstract classes achieves partial Abstraction

Interfaces on the other hand are used for 100% abstraction

Real life Example :

There is a parent class which defines Aircraft Body Construction rules

All child classes (Different Aircraft companies ) have to inherit parent classes and use those methods to build their own Aircraft.

But there is a method called Aircraft color which can be unique to every child class depending on their company logo.

So color method in parent class is not implemented with anybody. Instead only method is defined with out any body which we call as Abstract method.

If there is any method in class which is Abstract, then the class will be treated as Abstract class.

Unlike interfaces, this Abstract class can have methods (concrete) which have implementation as well. So, with Abstract class only partial Abstraction is achieved.

We cannot instantiate object for Abstract class. Child classes are forced to implement Abstract methods of Parent class.

Difference b/w Abstract classes & Interface :

Interface :

1. Interface contains only abstract methods
2. Access Specifiers for methods in interface must be public
3. Variables defined must be public,static,final
4. To implement an interface we use implements keyword

Abstract class:

1. Abstract class can contain abstract methods, concrete methods or both
2. Except private we can have any access specifier for methods in abstract class
3. Except private variables can have any access specifiers
4. To implement an Abstract we user extends keyword

There is a parent class which defines Artifact body construction rules like safety guidelines

So everything which are defined in one class. So all child classes, that means different aircraft companies like Emirates, Air India, Air France, American Airlines.

So we should make sure that all these aircraft companies should follow the rules defined by the parent class.

now in parent class, there is one abstract method called color. So this method do not have any implementation. So it's the duty of child class to inherit the parent class properties and methods with “extends” keyword and then you can have your own implementation of that color.So this abstract classes looks similar to interfaces

In interfaces, we have methods which do not have implementation, and it's the duty of classes to implement that interface and define your own methods.

But in interfaces, every method defined are abstract only.

So if a class have implemented methods and non implemented methods, then that class we called as a abstract class.

So, method which have body we say as concrete methods and non concrete methods are something which do not have body.

So these aircraft, guys have left one method to implement to your child classes

that made this aircraft parent classes abstract.

if you have only abstract methods means(with out the body), then that we would call as the interface interfaces achieve a hundred percent abstraction by hiding the implementation details,

if you have only abstract methods, then that we would call as the interface

create one new class( ParentAirCraft ) it is global to everyone, all the airlines.

I create one method called engine. So this method should follow some rules and regulations

so that all other aircraft will inherit this class and use this method to create engine

for their aircraft.

// why I’m adding the abstract word in the 1st line with class name ? – if there is any method in class which is abstract. Then the class will be treated as Abstarct class

Public abstract class ParentAirCraft{

// if u have some thing in ur braces then that is non-abstraction method

Public void engine()

{

Println(“follow Engine Guidelines”); // with body

}

Public void SafetyGuideLines()

{

Println(“follow safety Guidelines”);

}

Public abstract void bodycolor(); // method without body. This is abstraction method

create one new class( ChildEmirates )

So first of all, to inherit properties and methods from the parent class, you have to use inheritance concept and with a keyword called “extends”, you should be able to access all the methods and variables defined in this parent aircraft.

So when you are extending your parent aircraft, then it is mandatory for you to implement the methods which are left as abstract in your parent class. If you skip that, then your child class will not allow to execute, it'll throw compile error

Public class childEmirites extends ParentAirCraft{

Public static void main(String{} args){

ChildEmirites c =new ChildEmirites();

c.engine();

c.safeGuideline();

c.bodycolor();

//  if any class defined as an abstract, then it'll not allow you to instantiate that class.

So basically you cannot create object for the class which is marked as abstract.

( parentAirCraft p = new ParentAir Craft(); ) // if u create object it will throw error

}

Public void body color(){

Println(“Red color on the body”);

}

Note : 1. you cannot create object for the parent class which is marked as abstract.

2. Private is not allowed as access modifier for the methods defined in abstract class.(  every abstract class have to be inherited. So that's why they have avoided using private.)

282. Usage of inheritance in java

Vehicle - basic properties of vehicle are color,brakes,engine,gear.

Now, I am implementing new vehicle in my company. Let's say that is some new generation vehicle. So, already all the code are ready in vehicle parent class, and engine, brakes. So, it's already there in a parent vehicle class.

Hey, I have already code written to make the gear, engine, and brakes.If you want, you can use that."

// new generation vehicle like – increasing CC,RPM

Interface is something where there is no code in this main parent class. There are just defined signatures. We will take an agreement saying I will implement those, and then I would actually call them in my main class and write a code for it. But this is different.

Here, already code is written in one class. This is a class only. Let's say this is a parent class.

Now I want to use the properties and methods which are used in parent class in my class.

I am not going to write code or I'm not going to implement anything from this parent class.

I'll just use however it is in my parent class. That is inheritance.

Here I just copy paste it

So how would I use parent properties into my child class so that it is reusable ?

Create one new class(ParentClassDemo)

String colour =”red”;

Public void Gear()

{

Println(“gear code in implemented”)

}

Public void breaks()

{

Println(“breaks code in implemented”)

}

Public void audiosytem()

{

Println(“audiosystem code in implemented”)

}

Bring all this in to my child class

Createclass(ChildClassDemo)

actually, to get the properties of parent class to child class, I can do that by writing a keyword called extends. This is the keyword which helps us to inherit the properties from the parent.

// // So, this I acquired from my parent class, so I'm not defining it again because already some conditions are defined my parent vehicle, So, when I'm not modifying the color,

I would just simply bring it from my parent. But here, I'm modifying engine. That's the reason I've specifically written the engine method here

Public class childclassdemo extends parentclassdemo

Public void engine()

{

Println(“new engine”);

}

Public void color()

{

Prinltn(color); // if you not define the color also it will work bcze it will take it from parent clas

}

Public static void main (string[] args){

ChildClassDemo cd = new ChildClassDemo();

cd.colour();

cd.brakes();

283. Practical examples on types of inheritances

Note : Java does not allow multiple inheritances.( you cannot extend multiple parents.)

Class x

{

// class x memebers

}

Class y

{

//class y members

}

Class z extends x, y //Java does not allow multiple inheritances.( you cannot extend multiple parents.)

{

// class z members

}

Class A

{

Int i=10;

}

Class B extends A

{

Int i=20;

}

Public class mainclass

{

Public static void main(string[] args)

{

A a =new B();

System.out.println(a.i);

}

Output: 10

284.Inheritance code download

package coreJava;

public class childDemo extends parentDemo{  
  
String name ="QAClickAcademy";  
  
  
public childDemo()  
{  
super();// this should be always be at first line  
System.out.println("child class construtor");  
  
}  
public void getStringdata()  
{  
System.out.println(name);  
System.out.println(super.name);  
}  
  
  
public void getData()  
{  
super.getData();  
System.out.println("I am in child class");  
}  
public static void main(String[] args) {

childDemo cd = new childDemo();  
  
cd.getStringdata();  
cd.getData();  
}

}

// ParentDemo.js

package coreJava;

public class parentDemo {  
  
String name= "Rahul";  
  
public parentDemo()  
{  
System.out.println("Parent class construtor");  
  
}  
public void getData()  
{  
System.out.println(" I am parent class");  
}

public static void main(String[] args) {  
// TODO Auto-generated method stub

}

}

285. Explaining function overloading in java with example

what is function overloading?

So function overloading is basically using the same method name, multiple times,

but differentiating it based upon the arguments is called function overloading( in same class)

if they are differentiated only with the arguments, which are sending inside these methods,

then it is say that function is overloaded. That means they're having same names,

but they're differed in the arguments list.

So rules of function overloading is,

1. rules in having same method name in class was either argument counts should be different ( like in one method the argument is (int a); 2nd methods argument ( int a, int b);

2. argument data type should be different.

if you want to duplicate the method name, you have to make sure you follow these two rules.

package coreJava;  
//function overloading  
//either argument count should be different or  
//argument data type should be different  
public class childlevel extends childClassDemo {

public void getData(int a)  
{  
System.out.println(a);  
}  
public void getData(String a)  
{  
System.out.println(a);  
}  
public void getData(int a,int b)  
{  
System.out.println(b);  
}  
public static void main(String[] args) {  
// TODO Auto-generated method stub  
  
childlevel cl=new childlevel();  
cl.getData(2);  
cl.getData("hello");  
cl.getData(2, 5);

}

Ex : Real time : where we are using this functional overloading ?

When I am in an e-commerce website, you have purchase, you have selected something, add to cart and to your confirmation page. And there while confirming the payment,

there is a get payment method which calls. but one get payment method

takes debit card as an argument. Other get payment method takes credit card as an argument. And one payment might take gift card as an argument. So same get payment method only, but we are passing different kind of arguments. In that case, you could still stick to the get payment as a method name and pass different arguments and bring function overloading concept in that particular scenario.

286. How different function overriding from overloading?-Explain

So in the parent class we have gear, breaks, audio system.

so in my child class, so I decided to enhance my audio system.

So I thought of writing my own implementation for this new version of vehicle. Audio system.

New audio.

Same method(audio system) in both parent & child class.

cd.audiosystem();

if I execute this Whether it picks in my child class or in parent class? - It printed child one

So that means this audio system method in child class has overridden the audio system method present parent class. so when I am running this, it gives preference

to my local audio system method only. This is functional overriding.

In overriding, you will not see any difference. whereas in overloading, arguments are different.

So that's a big difference between overloading and overriding.

.. ChildDemo.java

package coreJava;

public class childDemo extends parentDemo{  
  
String name ="QAClickAcademy";  
  
  
public childDemo()  
{  
super();// this should be always be at first line  
System.out.println("child class construtor");  
  
}  
public void getStringdata()  
{  
System.out.println(name);  
System.out.println(super.name);  
}  
  
  
public void getData()  
{  
super.getData();  
System.out.println("I am in child class");  
}  
public static void main(String[] args) {  
// TODO Auto-generated method stub

childDemo cd = new childDemo();  
  
cd.getStringdata();  
cd.getData();  
}

}

288. What are Arrays and their usage in Java programmes

If I want to store multiple values into a single container, then we use arrays.

Int a=4; - I have defined a single variable called “a” and placed a value 4 inside that variable.

A container which stores multiple values of same data type.

Int a[] - this stands [] for, I'm taking multiple values.

Int a[ ] = new int[5]; -  keyword new basically allocates memory for an value in this array.

I'm saying that allocate five integer value space into this array a, that's it. That's how you define array.

arrayListexample ab=new arrayListexample();  
ab.abc();  
ab.abc();  
//ab.  
/\* int a;  
a=4;\*/  
  
//- A container which stores multiple values of same data type  
  
int a[] = new int[5];// Delcares an aray and allocates memory for the values  
a[0]= 2;  
a[1]=6;  
a[2]=1;  
a[3]=9;  
a[4]=12;//initilased values into that array

for(int i=0;i<a.length;i++)  
{  
System.out.println(a[i]);  
// retrieve values present in this array/  
}

We have successfully declared array and allocated memory for those complete array.

And we have initialized values into that array, and we are able to retrieve the values present

in that array.

So this is one of the method to actually declare array and retrieve the values.

289. Ways of initializing arrays and assigning objects

 If I want to declare one more array - int b [ ] = {6,9,8,7,3};

what is the difference between above highlighted 2 steps ?

So 1st one 'm actually allocating memory and then assigning the values.

2nd one , rather than allocating the memory I'm directly passing the values to this array. The memory is dynamically allocated here. memory is dynamically allocated. If you want in the same array you can add number bcze this dynamic we are not allocate memory constant.

So that's pretty much about single dimension array.

int b[] = {1,4,3,5,7,8};  
  
for(int i=0;i<b.length;i++)  
{  
System.out.println(b[i]);  
// retrieve values present in this array/  
}

Lets discuss multidimensional arrays now.

so each argument takes X axis and Y axis, X axis is a row, Y axis is a column.

So passing an objects in the both X axis and Y axis is nothing but multidimensional array.

So, to declare a multidimensional array, you need to provide one more bracket.

1st bracket represents x-axis or x row

2nd bracket represents y-axis or y column

int a[][] = new int[2][3];  
a[0][0]=2;  
a[0][1]=4;  
a[0][2]=5;  
a[1][0]=3;  
a[1][1]=4;  
a[1][2]=7;  
//System.out.println(a[1][0]);  
/\*int b[][]= {‌{2,4,5},{3,4,7},{5,2,1}};  
System.out.println(b[2][1]);\*/

291. Logic Program on Multi Dimensional arrays

We need two for loops here. One is for the row and one for column. So how the Java control has to move.

Req : 2 4 5

3 4 7

for(int i=0;i<2;i++) //row  
{  
for(int j=0;j<3;j++) //coloumn  
{  
System.out.println(a[i][j]);  
}

console : 2,4,5,3,4,7

So it starts from the row and complete all the items which are present in that row and then finishes off the row and next it starts for the second row and it completes all the items on that row with the columns.

292. Practise exercise - Print smallest number in 3\*3 matrix

Interview que : print minium number for the below matrix

2 4 5

3 4 7

1 2 9

I want the minimum number to print out of this multidimensional array matrix.

Create class(InterviewMinnumber).

Let me take array name as “abc” we are using multidimensional array that why I put double braces.

Fist we have declared an array int abc = {{2,4,5},{3,4,7},{1,2,9}};

Then We have to traverse each and every number on this three cross three matrix

and then come up with the smallest number.

Here we are using Two For Loops, outer For Loop for traversing the rows

inner For Loop, for traversing the columns. ( I,j is a variable ).

whatever is the zero row zero column. I would first assume that is the smallest and then I will traverse to each and every number, and then I compare. Whenever I extract number from it,

is that is less than the number what I have in my variable. If so, then go and swap.

Place that smallest number in my min variable.

Int min =abc[0][0]; // 0th row and 0th column

For(int i=0;i<3,i++)

{

For (int j=0;j<3;j++)

{

If(abc[i][j]<min) //2

{

Min=abc[i][j];

}

}

}

System.out.println(min);

293. Practise Exercise - Cisco Interview question on Arrays

Interview que :

wherever you get a minimum number. Let's say if I get a minimum number here, as zero in that column I need to get the maximum number.

2 4 5

3 2 10

1 2 0

That means 4 should be my output.

1. First identify the minimum number.
2. Identify/remember the column of min number
3. Find the max number in identified column

294. programming code download

int abc[][]={‌{2,4,5},{3,2,10},{1,2,0}};  
int min=abc[0][0];  
int mincoloumn = 0;  
  
  
for(int i=0;i<3;i++)  
{  
for(int j=0;j<3;j++)  
{  
if(abc[i][j]<min)//2  
{  
min=abc[i][j];  
mincoloumn=j;  
}  
}  
}  
//=1  
int max=abc[0][mincoloumn];  
int k = 0;  
while(k<3)  
{  
if(abc[k][mincoloumn]>max)  
{  
max=abc[k][mincoloumn];  
}  
k++;  
}  
  
System.out.println(max);  
}

}