Single Linked List Program:

import java.util.Scanner;

class Node {

int data; //instance variable

Node next; // instance variable of Node class

}

public class SingleLink {

static Node starter = null ;

static int get() {

return new Scanner(System.in).nextInt();

}

static int count() {

if(starter == null) {

return 0;

}else{

int count = 0 ;

Node temp ;

temp = starter ;

while(temp != null ) {

count++;

temp = temp.next;

}

return count;

}

}

void create() {

System.out.println("Enter number of nodes that u want to create.");

int no = get();

if(no > 0) {

Node temp = new Node() ;

starter = temp ;

System.out.println("Enter data.");

temp.data=get();

for(int i = 1; i< no ; i++) {

temp.next = new Node() ;

temp = temp.next ;

System.out.println("Enter data.");

temp.data = get();

}

temp=null;

}

System.out.println("Total nodes are : "+count());

}

void retrive() {

if(starter == null) {

System.out.println("No node is created.First u call create method.");

} else {

System.out.print("Data is : ");

Node temp ;

temp = starter ;

while(true) {

System.out.print(temp.data + "\t");

if(temp.next == null) {

break ;

} else {

temp = temp.next ;

}

}

}

System.out.println("\n\nTotal nodes are : "+count()) ;

}

void insert() {

if(starter == null) {

System.out.println("No node is created.First u call create method.") ;

} else {

System.out.println("Enter the position where u insert a node.");

int pos = get();

Node temp ;

temp = starter;

if(( pos > 1 ) && (pos <= count()+1 ) ) {

for(int i = 1 ; i < (pos-1) ; i++) {

temp = temp.next ;

}

Node temp1 ;

temp1 = temp.next ;

Node new1 = new Node() ;

System.out.println("Insert the data.");

new1.data = get();

new1.next = temp1 ;

temp.next = new1 ;

}

else if(pos == 1) {

Node new1 = new Node() ;

new1.next = starter;

starter = new1 ;

System.out.println("Insert the data.") ;

new1.data = get();

} else {

System.out.println("!!!!!Invalid Position.!!!!!") ;

}

}

System.out.println("Total nodes are : "+count()) ;

}

void delete() {

if(starter == null) {

System.out.println("No node is created.") ;

} else {

System.out.println("Press 1> for deleting the first node.") ;

System.out.println("Press 2> for deleting the last node.");

System.out.println("Press 3> for deleting any position.");

System.out.println("Enter the node position whom u delete.");

int pos = get() ;

if(pos == 1) {

Node temp ;

temp = starter ;

starter = temp.next;

temp.next = null;

temp = null ;

}

else if(pos==2) {

Node temp1,temp2;

temp1 = starter;

temp2 = null ;

while(temp1.next != null) {

temp2 = temp1 ;

temp1 = temp1.next ;

}

temp1 = null;

temp2.next = null;

}

else if(pos==3) {

System.out.println("Enter the position whom u delete.") ;

int no = get();

if(no > count()) {

System.out.println("!!!!!no Node is avaialable.!!!!");

}else{

Node temp1,temp2;

temp1 = starter;

temp2 = null ;

for(int i = 1 ;i<(no-1);i++) {

temp1 = temp1.next ;

}

temp2 = temp1.next ;

temp1.next = temp2.next ;

temp2 = null ;

System.out.println("Node is deleted.") ;

}

}else{

System.out.println("!!!!!Invalid Choice.!!!!");

}

}

System.out.println("Total nodes are : "+count()) ;

}

public static void main(String args[]) {

SingleLink ll = new SingleLink();

while(true) {

System.out.println("\t\t\t\*\*SingleLinkedList Application.\*\*");

System.out.println("\t\t\t1>Create nodes. ");

System.out.println("\t\t\t2>Traverse Nodes.");

System.out.println("\t\t\t3>Insert Nodes.");

System.out.println("\t\t\t4>Delete Node.");

System.out.println("\t\t\t5>Quit.");

System.out.println("Enter your choice.");

int ch = get();

switch(ch) {

case 1:

ll.create();

break;

case 2:

ll.retrive();

break;

case 3:

ll.insert();

break;

case 4:

ll.delete();

break;

case 5:

System.out.println("!!!!!Thank you for using my Application.");

System.exit(0);

default:

System.out.println("!!!!!Invalid Choice.!!!!!");

break;

}

}

}

}

Double Linked List Program:

import java.util.Scanner;

class Node

{

Node prev;

int data;

Node next;

}

public class Dll

{

public static Node start;

public static Node end;

public static int get()

{

Scanner sc=new Scanner(System.in);

return sc.nextInt();

}

public static int count()

{

int count=0;

if(start==null)

{

System.out.println("No node is available");

}else{

Node temp=start;

while(temp!=null)

{

count++;

temp=temp.next;

}

}

return count;

}

void create()

{

System.out.println("Enter the number of nodes you want to create : ");

int no=get();

if(no>0)

{

Node temp=new Node();

start=temp;

System.out.print("Enter the data value : ");

temp.data=get();

for(int i=1;i<no;i++)

{

temp.next=new Node();

temp.next.prev=temp;

temp=temp.next;

System.out.print("Enter the data value : ");

temp.data=get();

}

end=temp;

}

System.out.println("Total number of nodes are : "+count());

}

void traverse()

{

if(start==null)

{

System.out.println("No node is available");

}else{

Node temp=start;

System.out.print("Start -> ");

while(temp!=null)

{

System.out.print(temp.data+"-> ");

temp=temp.next;

}

System.out.print(" Last");

}

System.out.println("Total number of nodes are : "+count());

}

void delete()

{

if(start==null)

{

System.out.println("No node is available");

}

else if(start.next==null)

{

start=null;

end=null;

}else{

System.out.print("Enter the node/position you want to delete : ");

int pos=get();

if(pos==1)

{

Node temp=start;

start=start.next;

start.prev=null;

temp.next=null;

}

else if(pos==count())

{

end=end.prev;

end.next=null;

}else{

Node temp=start;

Node temp2=null;

for(int i=1;i<(pos-1);i++)

{

temp=temp.next;

}

temp2=temp.next;

temp.next=temp2.next;

temp2.prev=null;

temp2.next.prev=temp.next;

temp2.next=null;

}

}

System.out.println("Node deleted");

System.out.println("Total number of nodes are : "+count());

}

void insert()

{

if(start==null)

{

System.out.println("No node are available");

}else{

System.out.print("\nEnter the position where you want to insert : ");

int pos=get();

System.out.print("\nEnter the number of element you want to insert : ");

int no=get();

//Node temp=start;

for(int i=1;i<=no;i++)

{

Node temp=start;

if(pos==1)

{

start=new Node();

start.next=temp;

temp.prev=start;

System.out.println("Enter the data value : ");

start.data=get();

}

else if(pos==(count()+1))

{

Node mu=new Node();

System.out.println("Enter the data value : ");

mu.data=get();

end.next=mu;

mu.prev=end;

end=mu;

}else{

for(int k=1;k<(pos-1);k++)

temp=temp.next;

Node temp1=new Node();

System.out.println("Enter the data value : ");

temp1.data=get();

temp1.next=temp.next;

temp1.prev=temp;

temp.next.prev=temp1;

temp.next=temp1;

}

pos++;

}

System.out.println("Total number of nodes are : "+count());

}

}

public static void main(String args[])

{

Dll dl=new Dll();

while(true)

{

System.out.println("\n\n\n\t\tDOUBLE LINK LIST APPLICATION\n\t\t\*\*\*\*\*\*");

System.out.println("\t\t Chose your option\n\t\t \*\*");

System.out.println("\n\t\t1)Create Nodes\n\t\t2)Traverse Node\n\t\t3)Insert Nodes\n\t\t4)Delete Nodes\n\t\t5)Quit");

System.out.print("\n\tPlease yenter your choice : ");

int ch=get();

switch(ch)

{

case 1:

dl.create();

break;

case 2:

dl.traverse();

break;

case 3:

dl.insert();

break;

case 4:

dl.delete();

break;

case 5:

System.out.println("Thank you for using my application");

System.exit(0);

default:

System.out.println("Invalid option");

break;

}

}

}

}

\*Circular Linked List Program:

import java.util.Scanner;

class Node{

int data;

Node next;

}

public class Cll

{

static Node start;

static Node end;

public static int get()

{

Scanner sc=new Scanner(System.in);

return sc.nextInt();

}

public static int count()

{

if(start==null)

return 0;

int c=0;

Node temp=start;

do{

c++;

temp=temp.next;

}while(temp!=start);

return c;

}

void create()

{

System.out.print("Enter number of elements you want to store : ");

int pos=get();

Node temp=new Node();

start=temp;

System.out.print("Enter the data value : ");

temp.data=get();

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

{

temp.next=new Node();

temp=temp.next;

System.out.print("Enter the data value : ");

temp.data=get();

}

end=temp;

end.next=start;

System.out.println("Total number of nodes are : "+count());

}

void traverse()

{

if(start==null)

System.out.println("No nodes are avaliable");

else{

Node temp=start;

System.out.print("start -> ");

while(temp.next!=start)

{

System.out.print(temp.data+" -> ");

temp=temp.next;

}

System.out.print(temp.data+" -> last");

}

System.out.print("\nTotal number of nodes are: "+count());

}

void delete()

{

if(start==null)

System.out.println("No nodes are avaliable");

else if(start==end)

{

start=null;

end=null;

}else{

System.out.print("Enter the node number you want to delete : ");

int pos=get();

if(pos>count())

System.out.println("Invalid option");

else{

Node temp=start;

if(pos==1)

{

start=start.next;

temp.next=null;

end.next=start;

System.out.println("\nNodes are deleted");

}

else if(pos==count())

{

for(int i=1;i<(pos-1);i++)

{

temp=temp.next;

}

Node temp1=temp.next;

temp.next=start;

temp1=null;

System.out.println("\nNodes are deleted");

}else{

for(int i=1;i<(pos-1);i++)

temp=temp.next;

Node temp1=temp.next;

temp.next=temp1.next;

temp1.next=null;

System.out.println("\nNodes are deleted");

}

}

}

System.out.println("number of nodes are : "+count());

}

void insert()

{

if(start==null)

{

System.out.println("No nodes are avaliable");

}else{

System.out.print("Enter the position you want to insert : ");

int pos=get();

if(pos>(count()+1) && pos<1)

System.out.println("Invalid option");

else{

Node temp=start;

if(pos==1)

{

Node in=new Node();

in.data=get();

in.next=start;

start=in;

end.next=in;

}

else if(pos==(count()+1))

{

end.next=new Node();

end=end.next;

end.next=start;

System.out.print("Enter the data value : ");

end.data=get();

}else{

for(int i=1;i<(pos-1);i++)

temp=temp.next;

Node temp1=temp.next;

temp.next=new Node();

temp=temp.next;

System.out.print("Enter the data value : ");

temp.data=get();

temp.next=temp1;

}

}

}

System.out.println("Number of nodes are present : "+count());

}

public static void main(String args[])

{

Cll dl=new Cll();

while(true)

{

System.out.println("\n\n\n\t\tDOUBLE LINK LIST APPLICATION\n\t\t\*\*\*\*\*\*");

System.out.println("\t\t Chose your option\n\t\t \*\*");

System.out.println("\n\t\t1)Create Nodes\n\t\t2)Traverse Node\n\t\t3)Insert Nodes\n\t\t4)Delete Nodes\n\t\t5)Quit");

System.out.print("\n\tPlease yenter your choice : ");

int ch=get();

switch(ch)

{

case 1:

dl.create();

break;

case 2:

dl.traverse();

break;

case 3:

dl.insert();

break;

case 4:

dl.delete();

break;

case 5:

System.out.println("Thank you for using my application");

System.exit(0);

default:

System.out.println("Invalid option");

break;

}

}

}

}

Stack Application:

class Node{

int data;

Node next;

}

public class StackDemo

{

static Node top;

public static int getData()

{

java.util.Scanner sc=new java.util.Scanner(System.in);

return sc.nextInt();

}

public void push()

{

Node temp=new Node();

temp.next=top;

top=temp;

System.out.print("Enter the data value : ");

temp.data=getData();

}

public void pop()

{

if(top.next==null)

{

System.out.println("UnderFlow");

}else{

System.out.println("\n\nValue poped from Stack : "+top.data);

Node temp=top;

top=top.next;

temp.next=null;

}

}

public void display()

{

Node temp=top;

System.out.println("Currently present datas are : ");

while(temp!=null)

{

System.out.println("\t\t\*\*\*");

System.out.print("\t\t|\t");

System.out.print(temp.data);

System.out.println("\t\*\*\*");

temp=temp.next;

}

}

public void Peek()

{

System.out.println("Top data is : "+top.data);

}

public void replace()

{

System.out.println("Current value : "+top.data);

System.out.print("Enter new value : ");

top.data=getData();

}

public static void main(String muna[])

{

StackDemo ps=new StackDemo();

System.out.println("\t\t\*\* Welcome to muna's stack programm \*\*");

System.out.println("\t\t\t1->push()");

System.out.println("\t\t\t2->display()");

System.out.println("\t\t\t3->pop()");

System.out.println("\t\t\t4->peek()");

System.out.println("\t\t\t5->replace()");

System.out.println("\t\t\t6->EXIT");

while(true)

{

System.out.print("\t\tENTER YOUR CHOICE : ");

int ch=getData();

switch(ch)

{

case 1:

ps.push();

break;

case 2:

ps.display();

break;

case 3:

ps.pop();

break;

case 4:

ps.Peek();

break;

case 5:

ps.replace();

break;

case 6:

System.out.println("Thank you for using Muna's application");

System.exit(0);

default:

System.out.println("Invalid option !! Please choose a correct option");

break;

}

}

}

}

\*Queue Application:

class Node

{

int data;

Node next;

}

public class QueueDemo

{

static Node front;

static Node rear;

public static int getData()

{

java.util.Scanner sc=new java.util.Scanner(System.in);

return sc.nextInt();

}

public void push()

{

Node temp=new Node();

temp.data=getData();

if(front==null)

{

front=temp;

rear=temp;

}else{

rear.next=temp;

rear=temp;

}

}

public void display()

{

Node temp=front;;

System.out.print("start -> ");

while(temp!=null)

{

System.out.print(temp.data+" -> ");

temp=temp.next;

}

System.out.println("last");

}

public void pop()

{

Node temp=front;

if(temp==null)

{

System.out.println("!!!UnderFlow!!!");

}

else if(front==rear)

{

front=null;

rear=null;

}else{

System.out.println("Deleted data is : "+front.data);

front=front.next;

temp.next=null;

System.out.println("Deleted sucessfully");

}

}

public void replace()

{

System.out.println("Current value is : "+front.data);

System.out.println("Enter new value : ");

int d=getData();

front.data=d;

System.out.println("Replace Done");

}

public void peek()

{

System.out.println("Top data is : "+front.data);

}

public static void main(String swas[])

{

QueueDemo ps=new QueueDemo();

while(true)

{

System.out.println("\t\t\*\* Welcome to muna's stack programm \*\*");

System.out.println("\t\t\t1->push()");

System.out.println("\t\t\t2->display()");

System.out.println("\t\t\t3->pop()");

System.out.println("\t\t\t4->peek()");

System.out.println("\t\t\t5->replace()");

System.out.println("\t\t\t6->EXIT");

System.out.print("\t\tENTER YOUR CHOICE : ");

int ch=getData();

switch(ch)

{

case 1:

ps.push();

break;

case 2:

ps.display();

break;

case 3:

ps.pop();

break;

case 4:

ps.peek();

break;

case 5:

ps.replace();

break;

case 6:

System.out.println("Thank you for using Muna's application");

System.exit(0);

default:

System.out.println("Invalid option !! Please choose a correct option from muna's rull and ragulation");

break;

}

}

}

}

Q-1>What is java byte code?

Q-2>Describe java Buzzwords or features of java?

Q-3>Why java is a robust language?

Q-4>What is JVM?

Q-5>What is Unicode?

Q-6>What is this keyword? Uses of this keyword?

Q-7>What is super keyword? uses of seper keyword?

Q-8>What is method overloading? Uses of method overloading?

Q-9>What is method overriding? Uses of method verriding?

Q-10>What is dynamic method dispatch? uses of Dynamic method dispatch?

Q-11>How to declare constant in java?

Q-12>What is the use of transient variable?

Q-13>What is volatile variable?

Q-14>What is JNI? Uses of native method?

Q-15>Difference between final,finally and finalize?

Q-16>What is inner class? Types of inner classes in java? What is annonymous inner class?

Q-17>What is difference between abstract class and interface?

Q-18>What is command line arguments?

Q-19>What is Cosmic super class or Object class?

Q-20>What is Object Cloning?

Q-21>What is the difference between String and StringBuffer?

Q-22>What is marker interface?

Q-23>What is the difference between Error and Exception?

Q-24>What is the difference between throw and throws keyword?

Q-25>What is autoboxing and unboxing in java?

Q-26>What is Object Serialization?

Q-27>What is Externalization?

Q-28>What is reflection?

**Interview questions:**

Fundamental Concept:

Question1: What do you mean by Object Oriented Programming?

Ans-The birth of Object oriented programming can be traced to the 1960s.

Object oriented programming is simply a collection of object where object interact with each other through the method that have been embedded in them.

In object oriented programming the data and the operations to access and manipulate the data are closely entangled with one another.

Object oriented programming or OOP is the technique to create programs based on the real world.

Question2: What are 4 pillars of Object Oriented Programming?

Ans-The 4 pillars of Object oriented programming are: Abstraction, Encapsulation, Inheritance, and Polymorphism.

1-Abstraction:

Abstraction is defined as simplifying the complex reality of a problem. A programmer hides the complexity of the program from the end-user.

2-Encapsulation:

Encapsulation implies binding the information and methods to manipulate them together .This technique protects the data from outside interface and behaves as a protective wrapper.

3-Inheritance:

Inheritance is the mechanism to organize and structure software program .An inheritance facilitates code reusability. Due to implementation of this principle in OOPS, it is possible to create a child class which can inherit certain specific properties from its parent.

4-Polymorphism:

Polymorphism means one name many forms. In OOPs terminology it implies one method performing many tasks through which appropriate method invocation is performed by appropriate type specification.

Question3:Difference between procedural and Object Oriented language?

Ans-

Procedural Language:

In a procedural language, the instructions are executed sequentially.

It supports the preprocessor directives and header files.

In the procedural language, the programmer requires to allocate memory and to collect the garbage.

Object Oriented Language:

In Object Oriented Language, the total program is treated as a unit of object.

It does not have the preprocessor directives and header files, because predefined methods and classes are available in java through packages. Packages are the collection of class files.

In Object Oriented Language, the memory allocation and the garbage collection are done automatically.

Question4:What is the difference between parameters and arguments?

Ans-

Parameters:

These are the variables that receive the value from the method when it is called, then it is known as a parameter.

public int sum(int a,int b)

{

return a+b;

}

Here we return two int variables, those are parameters. Necessary conditions must be matched then the method is executed.

Argument:

When the value assign to the variable using the defined method is known as argument.

public int add()

{

a=10;

b=20;

return sum(a,b);

}

Here we return 10 and 20, those are arguments.

Question5: What's the difference between Queue and Stack?

Ans-

Queue: In java Queue is a predefined interface present in java.util package. Queue is the child interface of Collection. Queue typically order the elements in First in First out (FIFO).

Stack: In java Stack is a predefined class present in java.util package. Stack is the child class of Vector class. The object of the Stack class holds the elements that are in LIFO. A Stack executes the instruction in Last in First out (LIFO).

Question6: What is the purpose of a statement block?

Ans- Statement block is defined as to execute a sequence of statements which acts as a single group.

Question7:What are virtual function ?

Ans- In java, every non-static method is treated as virtual function. In every Object Oriented Language virtual function or virtual method is a function or a method which one is able to override in the child class. In java, when the method is static and final then the method is not treated as virtual method or virtual function. In C++, the functions are declared by virtual keyword is treated as virtual function. But, in java, virtual is not a keyword.

Question 8: What is reflection in java?

Ans-

Reflection in java allows Java code to invent the information for fields, constructors and methods of the loaded classes.

It also allows to call the fields, methods and constructors dynamically.

Reflection is the powerful feature of java.

It allows an executing java program to examine or to introspect upon itself and manipulate internal properties of program.

Question 9: What is JVM?

Ans-

JVM stands for Java Virtual Machine.

Java technology is based on the concept of Java Virtual Machine (JVM) that acts as a translator of byte code into machine code or JVM converts the Java byte code into platform specific machine language.

In java JVM not only translate byte code to machine code but also JVM manage memory, manages thread provide security to java’s byte code at the execution time.

Question 10:Why do we need public static void main(String args[]) method in Java?

Ans-

public – In java, main method is called by the JVM. If the main method is not defined through the public access specifier then JVM is not able to call the main method.

static – In java, static is a modifier. As the main method is static, JVM calls the main method through the class name without creating any instance of the class.

void – void is a keyword or return type of the main method.

main – main is the method from where execution will start. In java, main method is called by JVM.

String args[] – main method having only one argument i.e. array reference of String type.

Question 11: What if the static modifier is removed from the signature of the main method?

Ans- When the static modifier is removed from the main method then the java program compile successfully without generating any error message. But the program generates an Exception at the runtime named as: “NoSuchMethodError: main”.

Question 12: If I define the main method as “static public void main(String args[])” what happens?

Ans- Then the program compiles and execute successfully. We only check before the method name return type is present or not. But modifiers used to define main method do not depend on the order.

Question13: What is a void return type?

Ans-

void: If a method does not return anything then void is used to denote its return type during the definition of the method.

Question 14: What is the argument type of a program's main() method?

Ans- An array reference of String type is the argument type of the program's main() method in java.

Question 15: If the main method having no argument of array reference of String type then what happens?

Ans- Then the java source code compile successfully without generating any error but the program terminates at the runtime by generating an Exception at the runtime named as: “NoSuchMethodError: main”.

Question16: If I do not provide any arguments on the command line, then what is the size of the array?

Ans- In java when we pass command line arguments or not, then they are stored in an array of String type that is constructed by JVM at runtime. If we don’t provide any arguments on the command line then also JVM construct an array whose size is zero.

Question17:How can one prove that the array is not null but empty using one line of code?

Ans- In java, Array is a predefined class. As we know, Array elements are not only primitive data types but also Array elements are object references. If we want to extract the size of the array, we use length variable. In java, main method having only one argument i.e. array reference of String type. Whenever we pass the address of the array to the main method argument. When we are not passing the command line argument and print args.length then it prints zero but not prints null. So it indicates that Array is not null but empty.

Question 18: Can we define multiple main methods within a same class?

Ans- Within a single class we define multiple main methods. In java we are able to overload the main method.

Question 19: Can we import the same package or class twice? Will the JVM load the class twice at runtime?

Ans- Yes we can import the same package or class twice. How many times the package or class is imported but only once the class will load at runtime.

Question 20: How Java source code files are named?

Ans-

If we declare a class or interface with public access specifier, then the Java source code name is same as the class or interface name.

Within a source code file more than one public class or interface may not be present.

But when we declare a class or interface without using any access specifier then the java source code file must take the different name as compared to class or interface name.

We never declare the class and interface through private and protected access specifier.

.java extension is used by the source code files.

Question 21: What is platform?

Ans- In simple temp platform is a cross combination of hardware and software in which programmer executes the program.

Question 22: What is JIT?

Ans- Just-in-time compilation is also known as dynamic translation, is a method to improve the runtime performance of java byte code. JIT compilers represent a hybrid approach, with translation occurring continuously, as with interpreters but with caching of translated code to minimize performance degradation.

Question 23: What is java byte code?`

Ans- Byte codes are machine language of java virtual machine. When JVM loads a class file, it gets one stream of byte codes for each method in the class. The byte codes streams are stored in the method area of the JVM. The byte codes for a method are executed when that method invoked during the course of running the program. A byte code stream is a sequence of instructions for java virtual machine.

Question 24: What is Unicode?

Ans- Unicode is a system of encoding characters. All characters and Strings in java use the Unicode encoding, which allows truly international programming. Unicode is a universal international standard character encoding that is capable of representing most of the world’s written language. In Unicode characters hold 2 byte. Lowest value is ‘\u0000’ and highest value is ‘\uFFFF’.

Question 25: What is the difference between object and instance in java?

Ans- When we construct java object by calling the constructor then the object reference is initialized by JVM is called as instance. There is no difference between object and instance. In simple after object is constructed JVM when initialized the reference is treated as instance.

Variables in java:-

Question 26: What is the difference between a field variable and a local variable?

Ans-A field variable is a variable that is declared as a member of a class. In simple words field variable means class variable or static variable. If the field variable is not initialized then ClassLoader initialized it through the default value of data types. A local variable is a variable that is declared within a method, constructor and block. In java local variable must be initialized by the programmer before we use it.

Question 27: What is the difference between declaring a variable and defining a variable?

Ans-

Declaring a variable means describing the type and name of the variable, but at that time the variable is not initialized.

Defining the variable means at-a-time we describe the type and name of the variable and initialized it.

Question 28: What is the difference between static and non-static variables?

Ans-

static variable is treated as class variable, where as non-static or instance variable is treated as object variable.

static variables are initialized at the class loading time by the class loader, whereas an instance variable is initialized at the object creation time by the JVM.

There is only one copy of the static variable, but instance variables are initialized each time when new object is constructed.

Question 29: If a variable is declared as private, where may the variable be accessed?

Ans- If a variable is declared as private, then the variables may be accessed inside the class and the variables belonging to a class are not inherited to the subclass.

Question 30: What is a transient variable?

Ans-In java transient variables are not serialized. In java a static variable cannot declared as transient. Only instance variable is declared as transient.

Question 31: To what value is a variable of the boolean type automatically initialized?

Ans-The value of boolean type variable is automatically initialized to false. In simple boolean data type default value is false.

Question 32: Why we declare volatile variable?

Ans- In java volatile variable is used in multithreaded application. When multiple numbers of threads use the same variable then the thread will have its own copy of local cache for that variable. When one thread updates the value of the variable, then the updated value is stored in local cache but not in main memory. In multithreading the problem is avoided by volatile variable.

Datatypes in java:-

Question 33: Name the eight primitive data types.

Ans-The eight primitive data types are:

byte

short

int

long

float

double

char

boolean

Question 34: What is the range of the short type?

Ans-The range of short type is from -32768 to 32767.

Question 35: What is the range of the char type?

Ans-The range of char type is from “\u0000” to ” \uFFFF”.

Question 36: Is "abc" a primitive value?

Ans-No, “abc” is a String literal. “abc” is not a primitive value. It is a String object.

Blocks in java:-

Keywords and Identifiers in java:-

Question 37: Which characters may be used as the second character of an identifier, but not as the first character of an identifier?

Ans-Number’s may not be used as the first character of an identifier, but they may be used as the second character of an identifier.

Question 38: Is null a keyword?

Ans-Yes, null is a keyword.

It is a reference literal value. It can only be assign to reference variables. When a reference variable is declared as static and programmer not initialized it, then at the class loading time the ClassLoader initialized it through null value. Even if when a reference variable is declared as an instance variable, then at the object creation time JVM initialized it through null value.

Question 39: Explain the usage of the keyword transient?

Ans- The value of the transient variable are stored in memory but they are not stored in the persistent area like File or table. In simple words when the variables are declared as transient then object cannot be serialized.

Question 40: Is sizeof a keyword?

Ans-No, sizeof is not a keyword.

Question 41: Which non-Unicode letter characters may be used as the first character of an identifier?

Ans-The non-Unicode letter characters $ and \_ may appears as the first character of an identifier.

Question 42: Are true and false keywords?

Ans-Yes, true and false are keywords. Both are used as a boolean constant.

Command-Line Arguments:-

Wrapper class:-

Question 43: Describe the wrapper classes in Java?

Ans-Wrapper class is a wrapper around a primitive data type, which represents primitive data types in their corresponding class instances. For e.g. a boolean data type can be represented as a Boolean class instance.

Wrapper classes are broadly used with collection classes in the java.util package and with the classes in the java.lang.reflect reflection package.

Question 44: What are wrapped classes?

Ans-In java, primitive data types could be converted into object type by using the wrapper classes present in java.lang package.

Taking input from the keyboard:-

Question 45: What value does readLine() return when it has reached the end of a file?

Ans- It returns null, when it reached the end of a file.

JVM Architecture:-

Operator:-

Question 46: What is the difference between the >> and >>> operators?

Ans->>

It is a signed right shift operator.

The signed right shift (>>) operator shifts a bit (or bits) to the right by the distance specified in the right operand and fills the left most bit by the sign bit.

Syntax:

Variable>>number of shifts

>>>

It is a Bitwise unsigned right shift operator.

Bitwise unsigned right shift operator shifts the bits of a number to a specified number of bits towards right without preserving the signed bit.

Syntax:

Variable>>>number of shifts

Question 47: What are order of precedence and associativity?

Ans-The order of precedence and associativity are described below:

Question 48: What is the difference between the boolean & operator and && operator?

Ans-boolean & operator:

Syntax of boolean & operator:-

if(condition\_1 & condition\_2)

{

Codes

}

In & operator, both condition would have to be evaluated at-a-time.

boolean && operator:

Syntax of boolean && operator:-

if(condition\_1 && condition\_2)

{

Codes

}

In && operator, both condition\_1 and condition\_2 is checked. If both the parts are true, then codes within block are executed. If condition\_1 is checked as false then condition\_2 is never evaluated.

Question 49: Which Java operator is right associative?

Ans-In java the = operator is right associative

Question 50: What is the % operator?

Ans-It is a arithmetic operator known as modulus operator. This modulus operator is used to determine the remainder when a division is performed. The following example illustrates the operators.

Ex: int a=10;

int b=7;

System.out.println(a%b);//3

Question 51: Is the ternary operator written x : y ? z or x ? y : z ?

Ans-No, It is written in this way. Ex: x? y:z

Question 52: How is rounding performed under integer division?

ans:-In integer division, the factorial part of the result is truncated. This is known as rounding towards zero.

Question 53: Is &&= a valid Java operator?

Ans:-No, &&= is not a valid operator.

Question 54: What are the legal operands of the instanceof operator?

Ans:-The legal operands of the instanceof operator are an object and a class.

Object instanceof<class type>

Here object is the instanceof the class, where as class type is either the class name whose instance we pass or it is the base class.

Question 55: What is the difference between the prefix and postfix forms of the ++ operator?

Ans:-The difference between the prefix and postfix forms of the ++ operator is that, in a prefix expression, a value is incremented first, then the new value is restored back to the variable, where as in postfix expression, the current value is assigned to a variable first, then it is incremented by 1 and restored back to the original variable.

For Ex: int i=5;

i=i++;//5, this is postfix

i=++i;//6 this is prefix

Question 56: What is instanceof operator used for?

Ans:- Sometimes, knowing the type of an object becomes essential. For example programmer constructs a thread that generates various type of objects and construct another thread that processes the objects. In this situation it becomes essential for the processing thread to know the type of each object. Many invalid cast terminates the code at runtime by generating ClassCastException. So through the instanceof operator programmer collect the information about the object at runtime.

The syntax of the instanceof operator is given below, i.e.

Object instanceof <class type>

Here object is the instanceof the class, where as class type is either the class name whose instance we pass or it is the base class. This is an operator through which programmer got the information about the object at runtime.

.

Type casting:-

Question 57: What is casting?

Ans- Casting is used to explicitly convert the value of one type to another. It means the numeric values are converted from one numeric type to another or the object reference is changed to a compatible type by the cast operator. In java type casting is not only possible in case of primitive data types but also object references. Type casting of object references is only possible in case of inheritance.

Question 58: What is the default value of an object reference declared as an instance variable?

Ans-The default value of an object reference declared as an instance variable is null.

Question 59: Can a double value be cast to a byte?

Ans- Yes, double value can be casted to a byte type. Because in java, when a higher precision data type is casted with lower precision data type then explicit casting is done.

Example

public class Cast1

{

public static void main(String args[])

{

byte b;

double d=3.4;

b=d; //error

b=(byte)d;//Explicit Type casting

System.out.println(d);

}

}

Question 60: Can a Byte object be cast to a double value?

Ans- Yes, byte object can be casted to a double value. Because in java, when a lower precision data type is casted to a higher precision data type, then implicit casting is done.

Example:

public class Cast2

{

public static void main(String args[])

{

byte b1=67;

double d1;

d1=b1;

System.out.println(d1);

}

}

Question 61: How do you know if an explicit object casting is needed?

Ans-Instance of one class can be casted to instance of another class.

If you assign a super-class object to a variable of a subclass's data type, you need to do explicit casting.

An instance of the sub-class is cast when the instance of super class is assigned to the sub-class object.

Question 62: When can an object reference be cast to an interface reference?

Ans- A reference to an object can be cast into a reference of type interface if the class of the object implements the referenced interface. For Ex:

interface Test

{

void display();

}

public class Demo implements Test

{

public void display()

{

System.out.println(“See Java Program”);

}

public static void main(String args[])

{

Test tt=new Demo();

tt.display();

}

}

Question 63: What is numeric promotion?

Ans:-Numeric promotion means the numeric values are converted from one numeric type to another or the object reference is changed to a compatible type by the cast operator.

Looping:-

Question 64: Can a for statement loop indefinitely?

Ans:-Yes, a for statement loop can indefinitely. Because it is an entry control loop. The body of a loop of this type may not be executed at all if the specified condition fails at the start.

Question 65: What is the difference between a break statement and a continue statement?

Ans:-In java, both break and continue are used as a jump statement. But the big difference is:

break:

An early exit from a loop can be accomplished by using a break statement.

In case of nesting, loop break will exit only a single loop.

continue:

continue causes the loop to continue to the next iteration of the skipping statements.

In while and do loops, continue causes the control to go directly to test the condition, and then continue the process of iteration. In for loop, increment section of for is executed before the test condition is evaluated.

Question 66: How are commas used in the initialization and iteration parts of a for statement?

Ans:-In a for statement, commas are used to separate multiple statements within the initialization and iteration part.

i.e.

for (initialization; test condition; increment, or decrement)

{

Loop body;

}

Question 67: What is the difference between a while statement and a do statement?

Ans:-

while:

It is also known as entry control loop.

It is the control statement or control condition tested before the start of loop execution. The format of this loop is:

while (condition)

{

statement(s);

}

Here the body of the loop may not be executed at all if the condition is not satisfied for the 1st time.

do:

It is also known as exit control loop.

This is a control statement or a control condition which is placed at the end of the body of the loop. The format of this loop is:

do

{

statement(s);

} while (condition);

This loop will be executed till the condition is true.

Decision Making:-

Question 68: What restrictions are placed on the values of each case of a switch statement?

Ans:- The restrictions are placed on the values of each case of a switch statements are:

The expression inside the switch case may be an int or a char data type.

Each of these values should be unique in a switch block; otherwise it will hang the computer.

Each case must end with a colon.

The default statement is executed if there is no match with any case specified in the switch case.

Question 69: What is the difference between if statement and switch statement?

Ans:-

if:

It is a two-way decision making statement, which is used along with the construction of an expression.

Its syntax is:

if (expression)

{

statement(s);

}

statement (x);

The if statement of Java will only take the boolean value true or false.

switch:

Java has a built in multi-way design statement called a switch.

Its syntax is:

switch (expression/variable)

{

case Val 1:

statement(s);

break;

case Val 2:

statement(t);

break;

case Val 3:

statement(u);

break;

default:

statement(v)

break;

}

The expression inside the switch case may be an int or a char data types.

Class Fundamental:

Question 70: What is class?

Ans- Class is the collection of data members and member functions. Data members are nothing but simply a variable that we declare inside the class, so it call data members of that particular class. Member functions are the functions or methods which we declare inside the class is called as member function of that class. A class is a template for an object.

Question 71: What is Object?

Ans- Object is the instance of the class. In real world object share two characteristics: state and behavior. In simple an object store its state in fields or variables and exposes its behavior through methods.

Question 72: What is Singletone class?

Ans-

Singletone class is a class in which at a time only one object can create. This class constructor is private.

Within the Singletone class we define a static method for getting an instance of the Singletone class. But the Singletone class is instantiated only once. When a Singletone class is garbage collected and then reloaded then a new Singletone instance is created.

Through the following example Singletone class is elaborated.

public class Test

{

private static Test tt;

private Test()

{

System.out.println("Object of Singleton class");

}

public static Test getInstance()

{

if (tt == null)

{

synchronized(Test.class)

{

if (tt == null)

{

tt = new Test();

}

}

}

return tt;

}

public void display()

{

System.out.println("Check Object is Constructed");

}

public static void main(String args[])

{

Test aa=getInstance();

aa.display();

}

}

Question 73: What is a compilation unit?

Ans-A compilation unit is a source code file in java.

Question 74: What is encapsulation?

Ans- The wrapping of data and function together in a single unit is popularly treated as encapsulation. Encapsulation tells how to bind data. Through encapsulation we hide data. Bringing together data (attribute) and methods (behavior) within a single entity (class) is known as encapsulation.

Question 75: What is abstraction?

Ans- Abstraction is to represent essential features of a system without getting involved in the complexity of the entire system. Abstraction means simplifying the complex reality of a problem. For example cell phone users know only various functionalities available in cell phone without knowing how they are implemented.

Question 76: What is polymorphism?

Ans- Polymorphism means one name many forms. In OOPs terminology polymorphism implies, one method performing many tasks.

Question 77: Does java provide any predefined variable or method to find out the size of an object?

Ans- No, in java no predefined variable or method is present to find out the size of an object.

Question 78: What access level do you need to specify in the class declaration to ensure that only classes from the same directory can access it?

Ans-Through the default access level, we need to specify in the class declaration to ensure that only classes from the same directory/same package can access it.

Question 79:What is the difference between a public and a non-public class?

Ans-

1.public class:-

When a class is declared through the public access specifier is known as public class.

A public class can be invoked outside of the package.

2.non-public class:-

When a class is not declared with the public access specifier is known as non public class.

A non-public class cannot be invoked outside of the package.

Question 80: Can a lock be acquired on a class?

Ans-Yes, a lock can be acquired on a class.

Question 81: What are the field/method access levels (specifiers) and class access levels?

Ans-In java, there are 4 types of method access levels are present.

private:

This access specifier has the least visibility, i.e., it is only visible inside the class.

protected:

The variables and methods that are declared as protected can only be accessed through inheritance outside the package.

public:

This access specifier has the highest visibility, i.e., the methods or variables which are declared as public are visible to all.

no access:

If any access specifier is not specified, it is called no access or default access specifier.

In java, the methods are visible inside the entire package.

Question 82: Which class is extended by all other classes?

Ans-Object class is extended by all other classes. In java Object is the base class of all classes. Every user defined class implicitly inherited from Object class.

Question 83: If a class is declared without any access modifiers, where may the class be accessed?

Ans-If a class is declared without any access modifiers, then class by default takes no access specifier. That means the class only be accessed by other classes and interfaces that are defined within the same package.

Question 84: Which class should you use to obtain design information about an object?

Ans-We use the “Class” class to obtain design information about an object. In java Class is a predefined class declared in java.lang package.

Question 85: Is a class a subclass of itself?

Ans-Yes, a class is a subclass of itself.

Question 86: If a method is declared as protected, where may the method is accessed?

Ans-If a method is declared as protected, then the method is accessed within the class declared in the same package and through inheritance outside the package.

Question 87: What are the Object and Class classes used for?

Ans-In java, the Object class is the base class in the java class hierarchy. Every user defined class is implicitly inherited from Object class. Any user defined class can override Object class method. The class is used to represent all the codes, methods and blocks. One cannot write a single executable statement outside the class because the Java compiler would not permit it.

Question 88: Describe what happens when an object is created in Java?

Ans-

In java, we construct an object by calling the constructer explicitly by using new keyword. In java, object is constructed in 3 steps.

Declaration

Instantiation

Initialization

In declaration time, we declare a reference.

Syntax: <class name> reference;

In instantiation time, programmer explicitly calls the constructor of the class through new keyword.

Syntax: new <calling of constructor>;

In initialization time, JVM initialize the reference by assigning the base address of the object after the object is constructed.

Question 89: Can a top level class be private or protected?

Ans-No, we cannot declare a top level class be private or protected.

Constructor:-

Question 90: Can you call one constructor from another if a class has multiple constructors?

Ans- Yes, we can call one constructor from another if a class has multiple constructors through this keyword. But at the time of calling of constructor through this keyword, this must be the first statement within another constructor of the same class.

Question 91: What's the difference between constructors and other methods?

Ans-

Constructor:

A constructor is basically special method which is used to initialize a newly created object through a new operator.

It has the same name as the class in which it resides and is syntactically similar to a method.

In java we are bound to explicitly call the constructor.

A constructor cannot use any modifier.

Constructor cannot be override but constructor is overloaded.

Method:

A method in Java, is defined as a collection of statements which are grouped together to perform a particular operation.

Its name may or may not same as the class name in which it resides and is syntactically similar to a method.

We are bound to explicitly call the method.

Method can use any modifier.

Question 92: When does the compiler supply a default constructor for a class?

Ans-If no constructor is defined by the programmer, then the compiler automatically supply a default constructor for a class, when we call the default constructor.

They can be accessed directly inside the package, but accessed outside the package through inheritance.

this and super keyword:-

Question 93: How are this() and super() used with constructors?

Ans-Both this() method and super() method are used to call one constructor to another constructor of the same class.

this() is used to call one default constructor of the same class to another constructor. But this() must be the first statement.

super() is used in case of inheritance, when we want to call the base class constructor within the child class. But super() must be the first statement within the child class constructor.

Question 94: How are this and super used?

Ans-

this and super are the keyword in java.

Both are the non-static reference.

this keyword is implemented in the same class, where as super keyword is used in case of inheritance.

Use of this keyword:

When the instance variable name and the argument name is same then the instance variable name and the arguments name are same, then this keyword is used to differentiate between instance variable and the arguments.

When we want to call one constructor into another constructor of the same class then this keyword must be used. But this must be the first statement within the constructor at the time of calling another constructor.

Uses of super keyword:

When the base class instance variable name is same as child class instance variable name then super keyword must be used.

When child class constructor explicitly call the base class constructor within the child class constructor then super keyword must be used. But at the time of calling the constructor super must be the first statement within the constructor.

Inheritance:-

Question 95: How can a subclass call a method or a constructor defined in a super class?

Ans- In java, subclasses call the base class constructor and base class method by using super keyword.

Question 96: Does a class inherit the constructors of its super-class?

Ans-No, a class does not inherit the constructors of its super-class.

Question 97: Does java Support multiple inheritance?

Ans- Yes, java supports multiple inheritance by the help of interface. But java class not supports multiple inheritance.

Question 98:Does java support virtual functions?

Ans- No, java does not support virtual functions. But, every non-static method behaves like virtual function in java. In java, static and final method is not treated as virtual function.

Question 99: When a new object of derived Class is created, whose constructor will be called first, child’s or parent’s?

Ans- When the new object of derived class is created then the JVM/compiler must call the default constructor of the parent class and then the child constructor is called.

Method overloading and method overriding:-

Question 100: What is method overriding?

Ans- Method overriding is only possible in case of inheritance. Base class method is override in the child class. In case of method overriding method name is same, arguments of the method is same, return type of the method is same and access specifier of the method is same as base class method or least restricted than base class method. In java if base class method is static and final then the method cannot be override. Method overriding supports dynamic polymorphism.

Question 101: What restrictions are placed on method overloading?

Ans-Overloaded methods should have different parameter lists. We overload a method by defining same method name where as the arguments of the method is different.

Question 102: What restrictions are placed on method overriding?

Ans-Method overriding is only possible in the case of inheritance where super class method is overridden in its child class. In the case of method overriding, method name, its argument type, number of arguments, and return type is exactly the same. Access specifier may be the same or least restrictive in the child class where the parent class method is overridden. Private methods cannot be overridden. static and final methods can also not be overridden.

Question 103: What is dynamic method dispatch?

Ans- Dynamic method dispatch is only possible in case of inheritance. In case of dynamic method dispatch object refer to base class but object is constructed for child class. When the base class method is override in the child class then at the compile tile compiler check the signature of base class method where as at the runtime the override method is called or executed.

Inner class:-

Question 104: What are the different types of inner classes?

Ans- Inner classes is of four types.

static inner class

non static inner class

local inner class

anonymous inner class

Question 105: What modifiers may be used with an inner class that is a member of an outer class?

Ans-Inner class can be declared as static, abstract and final keyword and an inner class is declared by using any access modifier.

Question 106: Describe different types of inner classes?

Ans-A class defined within another class is treated as inner class.

In Java, four types of inner classes are used. These are: static inner class, non-static inner class, local inner class and anonymous inner class.

If an inner class uses the static modifier, then it is treated as a static inner class.

A class declared within another class without using a static modifier is treated as a non-static inner class. A non-static inner class is popularly known as an inner class.

When a class is declared within a method, it is treated as a local inner class.

One can also declare an inner class within the body of a method without naming it. These classes are known as anonymous inner classes.

Question 107: Can an inner class declared inside of a method access local variables of this method?

Ans- When a class is declared within a method, then the class is treated as local inner class. Local inner class is only invoke the final member of the method where the class is declared. So an inner class declared inside a method can’t invoked local variable of the method.

Question 108: Can an anonymous class be declared as implementing an interface and extending a class?

Ans-Yes, an anonymous class can be declared as implementing an interface and extending a class.

Question 109: What is the difference between a static and a non-static inner class?

Ans-

When a class is declared within another class through the static modifier, then it is treated as static inner class. But, when the class is declared within another class without using any modifier, then it is treated as non-static inner class.

A static inner class invoke the non-static member of the outer class through the outer class object, but a non-static inner class invoke a non-static member of the outer class directly without creating any instance of the outer class.

A static inner class never use “this” and “super” keyword, whereas a non-static inner class use “this” and “super” keyword.

Within the static inner class, we are able to declare a static member, but within a non-static inner class we are not able to declare a static member.

We are able to create the object of the static inner class within the main method or any static method, but we are not able to construct an object of the non-static inner class within the main method or static method.

Question 110: What modifiers can be used with a local inner class?

Ans-A local inner class is declared through a no-access modifier, but can’t be declared through public, private and protected access modifier.

Package:

Question 111: Explain the usage of Java packages.

Ans-Namespaces are called packages in Java.

Using packages is that it provides programmers with greater control over their source code.

Anyone can easily determine which classes and interfaces are related because in a package the classes and interfaces are grouped according to their functionality.

Question 112: What restrictions are placed on the location of a package statement within a source code file?

Ans-

Package name must be same as current directory name. Suppose the directory is e:\pack\p1. So the package name must be p1.

If the programmer does not declare the package then JVM automatically takes the package name as p1 and by default all the class files are stored in p1.

Question 113: In which package are most of the AWT events that support the event-delegation model defined?

Ans-The java.awt package and java.awt.event package are most of the AWT events that support the event-delegation model defined.

Question 114:Which package is always imported by default?

Ans-By default, java.lang package is always implicitly imported.

Question 115:What is a Java package and how is it used?

Ans- In java, package is a keyword, which encapsulates the class files. A package contains sub-packages and one or more than one class files.

Package is of two types;

1.Predefined package

2.User-defined package

For using the User-defined package, we have to follow three steps;

Step.1- Declaration of the package

Step.2- Import the class files resides in the package

Step.3- Declare the class

Question 116:Do we need to import java.lang package?

Ans- No, we don’t need to import java.lang package. As we know in java java.lang package is implicitly imported. Whatever classes and interfaces defined in java.lang package are automatically imported.

Modifiers:-

Question 117: What modifiers may be used with a top-level class?

Ans-In java top-level class is declared either through public or no access specifier.

Question 118: What is a native method?

Ans-

In java, native method is popularly known as foreign method. native methods are declared in java, but native methods are implemented either in C or C++.

In java, when the method declared through the native modifier, then it is treated as native method.

Question 119: When you declare a method as abstract method?

Ans- We declares a method as abstract, when we want to implement the properties of the method in the child class. Abstract method is declared in an abstract class but implement in the child classes.

Question 120: Can I call an abstract method from a non abstract method?

Ans-Yes, we can call an abstract method from a non abstract method.

Question 121: What is an abstract method?

Ans- When the method is declared through the abstract keyword then the method is treated as abstract method. Abstract method is only declared within the abstract class but abstract method is bound to be override in the child classes.

Question 122: Can an abstract class be final?

Ans-No, the final keyword cannot be used with an abstract class.

Question 123: Can you make an instance of abstract class?

Ans-No, we cannot make an instance of abstract class.

An abstract class cannot be instantiated.

Question 124: Can an abstract class have final method?

Ans-Yes, an abstract class can have final method.

Question 125: Can a final class have an abstract method?

Ans-No, a final class cannot have an abstract method.

Question 126: Give the example of four final classes?

Ans-

java.lang.String

java.lang.StringBuffer

java.lang.Array

java.lang.Math

Question 127: Give the example of four abstract classes?

Ans-

java.awt.Graphics

java.awt.Image

java.lang.Process

java.net.URLConnection

Question 128: What is the difference between an Abstract class and Interface in Java?

Ans-Difference between ‘abstract’ class and ‘interface’

abstract class interface

It never supports multiple inheritance. It supports multiple inheritance.

In abstract class, method may or may not be abstract. In interface each method is implicitly abstract.

Here the programmer defines a non-abstract method and constructor. Here, neither the constructor nor the method is defined by the programmer.

In abstract class programmer declares any type of variable. In interface whatever variables are declared, they are bound to be initialized as they are implicitly public static and final.

abstract class definitions begin with the keyword ‘abstract’ followed by class definition. An interface definition begins with the ‘interface’ keyword.

They are useful in a situation when some general methods are to be implemented and specialization behaviour are to be implemented by subclasses. Interfaces are useful in a situation when all its properties need to be implemented by the subclasses.

An abstract class can only have instance variables. All variables in an interface are by default public static final.

It may contain private as well as protected members. It can only have public members.

A class extending an abstract class need not implement any of the methods defined in the abstract class. A class implementing an interface must implement all of the methods defined in the interface.

If a new method is required to add in an abstract class, then it may simply be implemented and can also be called by its subclass. If a new method is required to add in an interface, then it must be implemented in all of the classes which implement that interface.

These are fast as compared to interface. These are slow as it requires extra indirection to find corresponding method in the actual class.

Interface:-

Question 129: What must a class do to implement an interface?

Ans- In java, we declare an interface for supporting multiple inheritance. As interface is implicitly abstract, interface cannot be instantiated. Within the interface whatever the methods are declared they are implicitly abstract. As every methods are implicitly abstract they are bound to be override in the child class. That is the reason for which a class implements an interface.

Question 130: What are null or Marker interfaces in Java?

Ans- The body of the marker interface is blank. But marker interface give the status to the programmer to perform a specific task. In java, Cloneable and Serializable is a marker interface.

Question 131: Why we should not have instance variable in an interface?

Ans-We should not have instance variable in an interface, because in interface whatever variables are declared, they are bound to be initialized as they are implicitly public, static and final.

Question 132: What modifiers may be used with an interface declaration?

Ans-We use public/no access modifier with an interface declaration.

Java.lang package:

Question 133: What is the purpose of the Runtime class?

Ans- Runtime is a predefined class present in java.lang package. A Runtime class is used to develop system level application.

Question 134: What is the purpose of the System class?

Ans- This class contains a large number of static methods and variables. One can neither create the object of a System class, nor a sub-class of this class.

System class contains several class fields and methods as System class cannot be instantiated.

System class provides the facility of standard input, standard output and error output streams, access to externally defined properties and environment variables.

Question 135: Which Math method is used to calculate the absolute value of a number?

Ans-The abs() method is used to calculate the absolute value of a number.

Question 136:Why are the methods of the Math class static?

Ans-The methods of the Math class is static because we can call the mathematical code library directly.

Question 137:What is a cloneable interface and how many methods does it contain?

Ans- The Cloneable interface, present inside the java.lang package, is used to create the clone of an object. The signature of Cloneable interface is:

public interface Cloneable

{

}

The Cloneable interface does not contain any method of its own. To clone an object, a class must implement the interface Cloneable and then invoke the clone() method of Object class. The signature of the clone() method is:

protected native Object clone() throws CloneNotSupportedException

According to the specification, the clone() method returns the reference of the Object class.

Garbage collection:

Question 138: Can an object's finalize() method be invoked while it is reachable?

Ans-No, an object's finalize() method be invoked while it is reachable.

Question 139: What is the purpose of finalization?

Ans-The purpose of finalization is to called the finalize() method before the garbage collector releases the memory occupied by an object.

Question 140: How many times may an object's finalize() method be invoked by the garbage collector?

Ans--An object's finalize() method may be invoked by the garbage collector once.

Question 141: Explain garbage collection?

Ans. In some of the object-oriented languages it is required to keep track of all the objects that are created and to destroy them explicitly when they are no longer needed. Hence memory management has to be done explicitly, which is tedious and error-prone. This problem is overcome by Java platform, which allow to create as many objects as per requirement and the Java runtime environment deletes the objects when it determines that they are no longer being used. This process is called garbage collection.

Question 142: How can you force garbage collection?

Ans- No, we can’t force garbage collection .but we can request it by invoking System.gc() method and Runntime.getRuntime.gc(); method.

Question 143: Does garbage collection guarantee that a program will not run out of memory?

Ans:-No, garbage collection does not guarantee that a program will not run out of memory.

Question 144: When is an object subject to garbage collection?

Ans:-When the garbage collection becomes unreachable to the program, then an object is subject to garbage collection.

Question 145: How can you minimize the need of garbage collection and make the memory use more effective?

Ans:-By the help of object pooling and references of weak object we can minimize the need of garbage collection and make the memory use more effective.

Question 146: Under what conditions is an object's finalize() method invoked by the garbage collector?

Ans-Before an object is garbage collected, garbage collector implicitly calls finalize() method of object class.

import java.util.Scanner;

public class Add

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter first number");

int a=ss.nextInt();

System.out.println("Enter second number");

int b=ss.nextInt();

int sum=0;

for(int i=0;i<b;i++)

{

a++;

}

System.out.println("sum of two number is : " +a);

}

}

//Enter a four digit number and add 1st and 4th digit

import java.util.Scanner;

public class AddDigit

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter a number");

int num=ss.nextInt();

int i=0, temp=0,sum=0;

while(num>0)

{

temp=num%10;

if((i==0) || (i==3))

{

sum=sum+temp;

}

num=num/10;

i++;

}

System.out.println("sum= "+sum);

}

}

import java.util.Scanner;

public class Binary

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter a number");

int num=ss.nextInt();

int b=0,v=0;

while(num>0)

{

if((num%10==0) || (num%10==1))

{

b++;

}

v++;

num=num/10;

}

if(b==v)

{

System.out.println("yes its a binary number");

}else{

System.out.println("No its not a binary number");

}

}

}

import java.util.Scanner;

public class CountDigit

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter a number");

int num=ss.nextInt();

int r,sum=0;

while(num!=0)

{

r=num%10;

sum++;

num/=10;

}

System.out.println("Total digit is : "+sum) ;

}

}

public class Divisor

{

public static void main(String args[])

{

int max=0,count=0,i,j,k=0;

for(i=1;i<=100;i++)

{

//i=1;

for(j=1;j<=i;j++)

{

if(i%j==0)

{

count++;

}

if(count>max)

{

max=count;

k=i;

}

}

System.out.println(max+"\t"+k);

}

}

}

import java.util.Scanner;

public class GcdLcm

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter first number");

int a=ss.nextInt();

System.out.println("Enter second number");

int b=ss.nextInt();

int m=a\*b;

while(a!=b)

{

if(a>b)

{

a=a-b;

}else{

b=b-a;

}

}

int lcm=m/b;

System.out.println("GCD = "+b+"\tLCM = "+lcm);

}

}

/\*

1

1 1

1 2 1

1 3 3 1

\*/

import java.util.Scanner;

public class pascal

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter number of rows");

int row=ss.nextInt();

int a=1,p,q=0,b;

for(q=0;q<row;q++)

{

for(p=30-3\*q;p>0;p--)

System.out.print(" ");

for(b=0;b<=q;b++)

{

if(b==0 || q==0)

{

a=1;

}else{

a=(a\*(q-b+1)/b);

}

System.out.print(" "+a);

}

System.out.println();

}

}

}

import java.util.Scanner;

class node

{

int power[]=new int[Polynomial.n];

int coef;

node next;

}

class Polynomial

{

static int n;

static int input()

{

return (new Scanner(System.in).nextInt());

}

static int pow(int x,int a)

{

if (a==0)

return 1;

int r=1;

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

r\*=x;

return r;

}

public static void main(String args[])

{

System.out.println("\n\n\tCalculating POLYNOMIAL Function using Single Linked List.");

System.out.println("\t==========================\n\n");

System.out.print("Enter number of variables : ");

n=input();

node start;

node temp=new node();

start=temp;

int x[]=new int[n];

System.out.println("1st enter Coefficient of variable & then enter its Power.");

System.out.println("It STOPs automatically when all variables power becomes 0");

int t=1;

while(true)

{

System.out.print("\n\n\tCoefficient of term "+t+" : ");

temp.coef=input();

for(int i=0;i<n;i++)

{

System.out.print("\t Power of x "+(i+1)+" : ");

temp.power[i]=input();

}

int sum=0;

for(int i=0;i<n;i++)

sum=sum+temp.power[i];

if(sum<=0)

break;

System.out.print("\nAny more term?(y / n) : ");

char ch=new Scanner(System.in).nextLine().charAt(0);

if(ch=='n'||ch=='N')

break;

temp.next=new node();

temp=temp.next;

t++;

}

for(int i=0;i<n;i++)

{

System.out.print("\n Enter value of variable x "+(i+1)+" : ");

x[i]=input();

}

temp=start;

int sum=0;

while(temp!=null)

{

int tot=1;

for(int i=0;i<n;i++)

tot=tot\*pow(x[i],temp.power[i]);

sum=sum+(temp.coef\*tot);

temp=temp.next;

}

System.out.println("\n\nSUM OF POLYNOMIAL FUNCTION : "+sum);

}

}

import java.util.Scanner;

public class PositiveInteger

{

public static void main(String args[])

{

Scanner ss=new Scanner(System.in);

System.out.println("Enter a number");

int num=ss.nextInt();

for(int i=2;i<num;i++)

{

if(num%i==0)

{

System.out.println(i);

}

}

}

}

public class WithoutSemicolon

{

public static void main(String[] args)

{

if(System.out.printf("java program")!=null)

{

}

}

}