नहि ज्ञानेन साधुश्रमम्

ಶ್ರೀ ಮೇಧಾ ಪದವಿ ಮತ್ತು ಪದವಿ-ಪೂರ್ವ ಮಹಾವಿದ್ಯಾಲಯ

SHREE MEDHA DEGREE AND PRE-UNIVERSITY COLLEGE

ISO 9001-2015 Certified College

#3 Fort Rd, Ballari

DEPARTMENT OF COMPUTER SCIENCE

PREVIOUS YEAR's Q&A

J A V A

Mr. Chaitanya Reddy S V

2021

2M QUESTION & ANSWER

01. Write a syntax and example of creating array in java? [2019]

Ans.

Syntax: - datatype array_name[size];

Example: - int a [10];

02. Define string with example [2019]

Ans.

Sequence of characters is known as string.

Example: - "Hello" is a string of 5 characters

03. Who invented java and when? [2019]

Ans.

James gosling and his team members developed java programming Language in sun microsystem company of USA based in the year 1991.

04. What are the different types of polymorphism? [2019]

Ans.

There are two types of polymorphism: -

1. Compile time polymorphism.
2. Run time polymorphism.

05. What is final keyword? [2019]

Ans.

In java the final keyword can be used while declaring an entity using the final keyword means that value can't be modified in the future.

06. Mention purpose of defining get document base () and get code base ()? [2019]

Ans.

Get document base(): - this method is used to return the URL of the directory in which the document is resides.

Get code base(): - this method is used to establish a path to other files (or) folders tha t are in the location as the class being run.

07. What is applet? Name two types of the applet? [2019,2017]

Ans.

Applet is a small application that is embedded in a HTML page which is accessed an dtransported over the internet and automatically installed into the client machines and run as a part of web page

Types of applet: -

- AWT (abstract window toolkit)
- JApplet class

08. Define event source and event listener? [2019]

Ans.

Event source: - A source is an object that generates an object. This occurs when object changes in some way.

Event listener: - Event listeners represent the interfaces responsible to handle events java provides various event listener classes.

09. List any two event handling classes? [2019]

Ans.

Event handling classes are: -

- Mouse event.
- Key event.
- Window event.
- Item event.
- Text event.
- Action event.

10. Define thread? [2019]

Ans.

- Thread is a part of program that is running.
- A thread is a single sequential how of control within a process.
- A thread also referred is a light weight process.

11. Explain history of java? [2018,2017]

Ans.

- The history of java starts with green team.
- The principles of creating java programming are simple, robust, portable, secured, multithread etc
- Java was developed by James gosling who is known as the father of java.
- In 19995 James gosling and his team members started the project.
- Currently java is used in internet programming.

12. Write the general syntax for defining data type? [2018]

Ans.

Java is statically typed language. The base of any programming language is its data type and variables.

- Data type specify the size and type of values that can be stored.java language is rich in its data types.
- The verities of data types available allows the programmer to select the type appropriate to the needs of the application.
- There are two categories of data types in java:
  1. primitive data type
  2. non-primitive data type

13. Explain JVM, AWT, URL, JSL and API? [2017,2018]

Ans.

JVM- Java Virtual Machine.

- Jvm is machine which executes the java program.
- Jvm is a heart of java program execution process.
- Jvm is a very important part of both JDK and JRE because it is contained or inbuilt in both.

AWT- Abstract Window Tool.

- Is a set of application program interfaces (API) used by java programmers to create graphical user interface (GUI) objects, such as buttons, scroll bars and window.
- AWT is a part of the java foundation classes (JFC) from sun micro-systems, the company that originated java.

URL- Uniform Resource Locator

- Uniform resource locator and represent on the world wide web, such as web page or FTP directory.
- This section shows you how to write java programs that communicate with URL.

JSL-Java Search Library

- Java search library or java simulation library.
- JSL is a library written in java that provides a frame work for general searching and graphs.

API- Application Programming Interface.

- API includes hundreds of classes and methods which are grouped into several functional packages.
- some of the most commonly used packages are:
  1.language support package.
  2.utilities package.
  3.input and output package.
  4.networking package.

14. What are the logical and relational operators? [2018]

Ans.

Logical operator- logical operator is an addition to the relational operator. java has three logical operators which are given in a table

Operators	Meaning
&&	logical AND
||	logical OR
!	logical NOT

- The logical operators && and || are used when we want to form compound conditions by combining two or more relations.
- eg: a > b && x == 10.

Relational operator:

- We often compare two quantities, and depending on their relation, take certain decisions.
- For example: We may compare the age of two persons, or the price of two items, and so on. this comparison can be done with the help of relational operators. We already used the symbol "<" meaning "less than". an expression such as

  a < b or x < 20

- eg: if x=10, then
  x < 20 is true
  while
  20 < x is false
- Java supports six relational operators in all. these operators and their meainigs are shown in below table

operator	meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
= =	is equal to
! =	is not equal to

15. Define packages. What are the java API packages? [2018]

Ans.

Java package organizes java classes into name spaces, providing a unique namespace for each type it contains.

- classes in the same package can access each other's package of private and protected members.
- The java packages are classified into two categories
  1. java API packages.
  2. user defined packages.

  Java API packages: -
- Java API provides a large number of classes grouped into different packages according to the functionality.
- Most of the time we use the package available with the java API.
- The functional breakdown of package that are frequently used in the programs.

16. Define stopping and blocking of thread? [2018]

Ans.

Stopping a thread: -

- whenever we want stop a thread from running further, we may do so by calling its STOP() method like : a thread.stop();
- This statement causes the thread to move to DEAD state. A thread will also move to the dead state automatically when it reaches the end of its method. The STOP () method may be used when the premature death of thread desired.

  Blocking a thread: -
- A thread can also be temporarily suspended or blocked from entering into the runnable and subsequently running state by using either of the following thread methods:

  Sleep() //blocked for a specified time
  Suspend() //blocked until further orders
  Wait() //blocked until certain conditions occurs.

  These methods cause the thread to go into the **Blocked** (or **not runnable**) state this thread will return to runnable state when the specified time is elapsed in the case of **Sleep**(), the

Resume() methods is invoked in the case of **Suspended** () and the **Notify**() method is called in the case of **Wait**().

17. Explain syntax of Exceptional handling? [2018]

Ans.

The java mechanism that deals with handling the errors in an organised fashion is called as exception handling.

- The basic concepts of exception handling are throwing an exception and catching it.
- Java use a keyword **try** to prefer a block of code that is likely to cause an error condition and "throw" an exception. A catch block defined by the keyword **catch** "catches" the exception "thrown" by the try block and its appropriately. The catch block is added immediately after the try block.

SYNTAX: -

```
try
{
        // do risky things;              //generate an a exception
}
 catch [exception ex]
{
          // try to recover;            //processes the exception
}
```

18. Define Final class of Final method? [2018]

Ans.

A class that cannot be sub classed is called a final class. or the class which cannot be inherited is called has a final class.

example: - Method overloading.

SYNTAX:

final class Aclass(.....)

final class Bclass extends someclasses(.....)

Final method: -

- The final method cannot be over ridden, to avoid the sub class to override the method we can use final method.

19. Define passing parameter to applet? [2018]

Ans.

Passing parameters to an applet code using <**PARAM**> tag is something similar to passing parameters to the **main**() method using commend line arguments. To set up and handle parameters we need to do two things:

1. Include appropriate <**PARAM**...>tags in the html document

2.Provide code in the applet to parse these parameters.

- Parameters are passed on an applet when it is loaded. We can define the **init**() method in the applet to get hold of the parameters defined in the <**PARAM**> tags this is done using **getparameter**() method. Which takes one string arguments representing the name of the parameter and returns a string containing the value of the parameter.

SYNTAX: -

<PARAM NAME = text value = " I LOVE JAVA">

20. Define event handling? [2018,2017,2016]

Ans.

As name suggest, event handling is a mechanism that is used to handle event generated by applets. A event could be the accurance of any activity such as mouse click or a key press.

Some the key events in java are

1. action event
2. item event
3. text event
4. window event
5.key event

21. Define class. Write the general syntax for defining Class? [2017]

Ans.

A class is a template that defines the form of an object.

- An object is an instance of a class.
- When we define a class, we declare its exact form and nature and we do this by specifying the instance variable that it contains and the method that operate on them.
- Although very simple classes might contain only method or only instance variables, most real word classes contain both.
- A class is created using a keyword "class".
- The general form of class definition is class

SYNTAX: -

```
class name
{
 data members; // variables
        data methods; // function
}
```

22. What is bitwise operators and special operator? [2017]

Ans.

java has a distinction of supporting special operators known as bitwise operators for manipulation of data at values of bit level. These operators are used for testing the bits, or shifting them to the right or left . Bitwise operators may not be applied to **float** or **double**.

- The list of the bitwise operators

operator	meaning
&	Bitwise AND
!	Bitwise OR
^	Bitwise EXCLUSIVE OR
~	One's complement
<<	Shift left
>>	Shift right
>>>	Shift right with zero fill

Special operators: -

- Java supports some special operators of interest such as INSTANCEOF operator and member selection operator (.).
- There are two types of special operator they are:

1.Instanceof operator

2.Dot operator

23. Define constructor with an example? [2017]

Ans.

A constructor initializes an object when it is created and it has the same name as its class and is syntactically similar to the method.

- We know how to create an object of a class, if we want to set the default values for instance variables at the time creation of an object then we should constructors.
- There are two types of constructors they are:

1.Default constructors.

2.Parameterized constructors.

example:

```
class test
{
        test ()
{
    //constructor body
}
}
```

Here, test() is a constructor. It has the same name as that of the class does not have a return type.

24. What is an exception handling mechanism? [2017]

Ans.

Exception handling is a mechanism to handle runtime errors such as class NotfoundException, IOException, SQLException, RemoteException, etc.

25. Define thread, how to create a thread in java? [2017]

Ans.

A thread is a part of a program that is running.

- A thread is a signal sequential flow of control within a process.
- A thread is also referred to as light weight process.

Creating threads: -

Creating threads in java is simple. threads are implemented in the form of objects that contain a method called **run**(). The **run**() method is the heart and soul of any thread. It makes up the entire body of a thread and is the only method in which the threads behaviour can be implemented. A typical **run**() would appear as follows

```
public void run()
{
    .....
```

..... (statements for implementing thread)
.....
}

26. What is abstract method and abstract class? [2017]

Ans.

The method which does not have a body is called as an abstract method but that only contain signature.

•The method can never be sub class java allows us to something that is exactly opposite to this i.e., we can indicate that a method must always be redefined in a sub class, thus making overriding compulsory. This is done using modifier key word ABSTARCT in the method definition.

•while using abstract class we must satisfy the following conditions

1. We cannot use abstract classes to instantiate objects directly. for example:

shape s = new shape()

Is illegal because SHAPE is an abstract class.

2. The abstract method of an abstract class must be defined in its sub class.
3. We cannot declare abstract constructors or abstract static methods.

for example: -

```
abstract class shape
{
    .....
    .....
  abstract void draw ();
  .....
  .....
}
```

27. Define operator. list out different operators supported by Java? [2016]

Ans.

Operator is a symbol that the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. They usually form a part of mathematical or logical expressions.

- java operators can be classified into a number of related categories as below;

1. Arithmetic operators
2. Relational operators
3. Logical operators.
4. Assignment operators
5. Increment and Decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators.

28. What is final variable and final method? [2016]

Ans.

Final variable: -

- The value of the variable declared as a final cannot be changed once a value is assigned is called final variable.

Final method: -

- The final methods cannot be overridden, to avoid the sub class to override the method we can use final method is called final method.

29. Define interface. How it differs from class? [2016]

Ans.

An interface a programming construct which contains only public abstract methods and public static final variables.

- A interface is not extended by a class it is implemented by a class.
- The difference between class and interface is:

Class	Interface
1.The members of class can be declared as constant or variables.	1.The members of an interface are always i.e. their values are final.
2.It can be instantiated by declaring objects	2.It cannot be used to declare objects. It can only be inherited by a class
3.It can use various access specifiers like public private, or protected.	3.It can only use public access specifiers.

30. Define constants. Give the general syntax of declaring constant in java? [2016]

Ans.

Constants: constant in java referred to fixed values that do not change during the execution of the program.

SYNATX:

static final datatype indetifier_name = constant;

- The static modifier causes the variable to be available without an instance of its defining class being loaded.
- The final modifier makes the variable and changeable.

31. Describe different methods of audio clip Interfaces? [2016]

Ans.

The AudioClip interface defines these methods: play( ) (play a clip from the beginning(), stop ( ) (stop playing the clip), and loop( ) (play the loop continously().

- After you have loaded an audio clip using getAudioClip( ),you can use these methods to play it.
- once an audio file is loaded into memory with getAudioClip( ),you use the AudioClip interface to work with it.

32. Define thread. give the general syntax of creating thread? [2016]

Ans.

A thread is a part of a program that is running.

- A thread is a singal sequential flow of control within a process.
- A thread is also reffered to as light weight process.

Creating threads: -

Creating threads in java is simple. threads are implemented in the form of objects that contain a method called **run**(). the **run**() method is the heart and soul of any thread. it makes up the entire body of a thread and is the only method in which the threads behaviour can be implemented. a typical **run**() would appear as follows

```
public void run()
{
        .....
..... (statements for implementing thread)
.....
}
```

33. Name any two event classes? [2016]

Ans.

All the events in java have corresponding event classes associated with them. each of this classes is derived from one single super class that is **eventobject**.

- It is contained in the **java.util package**.
- The **eventobject** class contains the following two important methods for handling events:
  1. **getSource**():return the event source.
  2. **toString**():returns a string containing information about the event source.

34. What is adapter class? [2016]

Ans.

Java adapter classes provide the default implementation of listener interface. if you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interface.

- The adapter classes are found in **java.awt.event**, **java.awt.dnd** and **javax.swing.event** packages.

5M QUESTION & ANSWER

01. Write short note on jdk [2019]

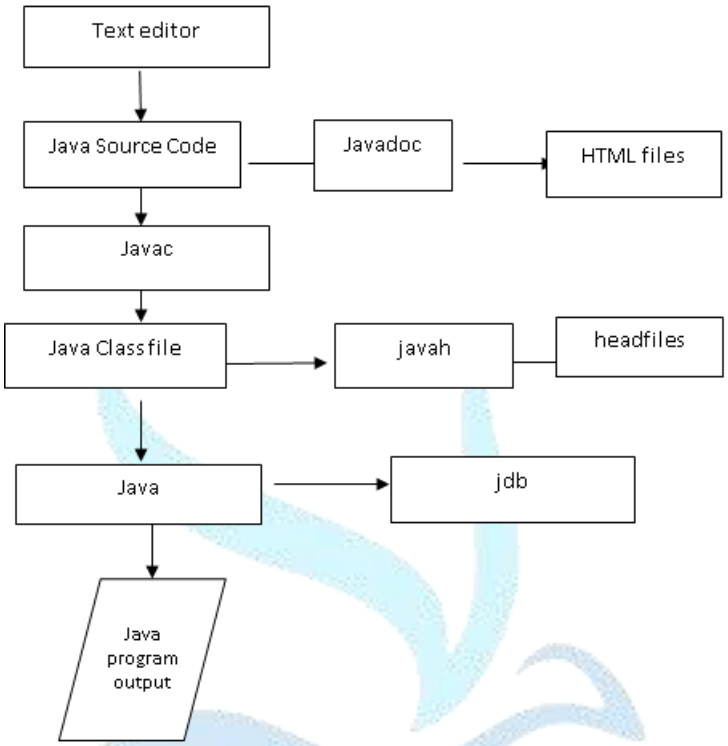
Ans.

The Java Development Kit (JDK) is a program development environment for writing applets and applications.

JDK comes with a collection of tools that are used for developing and running Java programs.They include:

- appletviewer(For viewing Java Applets)
- javac(Java compiler)
- java(Java interpreter)

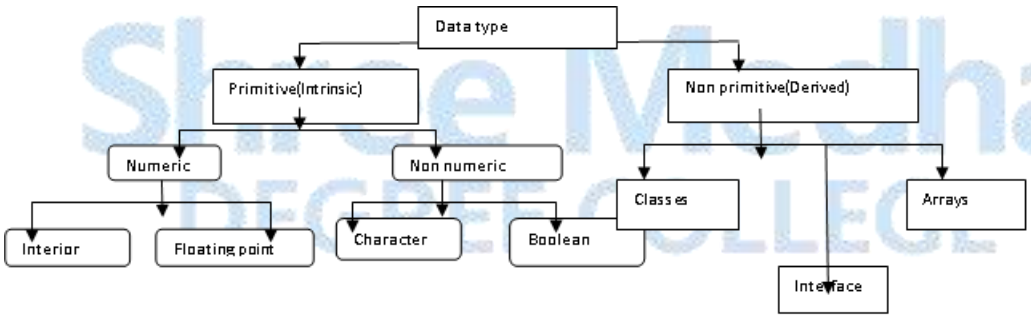
- javap(Java disassemble)
- javah(For c header file)
- javadoc(For creating HTML documeents)
- jdb(Java debugger)



02. List the different datatypes in java and illustrate through example [2019]

Ans.

Every variable in Java has a data type.Data types specify the size and type of values that can be stored.



INTEGER TYPE: Integer types can hold whole numbers. Java supports four types of integers. They are byte, short, int and long.

1.**BYTE**:

- The size of byte is 1 byte.
- The minimum value is -128 and maximum is +127.
- The default value is zero (0).
- It is used to space in large array mainly in space of integers since a byte is four time smaller than int.

Example: byte a=100, byte b=-50.

2.**SHORT**:

- The size of short is 2 bytes.
- The minimum value is -32768 and the maximum value is _32767.
- The default value is zero (0).
- It is used to save memory same as byte data type because a short data type is a two times smaller than it.

Example: short s=10000, short r=-20000.

3.**INT**:

- The size of the int is 4 bytes
- The minimum value is -2, 147, 483, 648 to maximum value is +2,147,483,647.
- The default value is zero (0).
- It is generally used as default data type for intergeral values unless there is a concern about memory.

Example: int a=100000 , int b=-200000.

4.**LONG**:

- The size of long is 8 bytes.
- The minimum value is -9,223,372,036,854,775,808 and minimum value is +9,223,372,036,854,775,807.
- The default value is zero (0).
- It is used when wider range of int is needed.

Example: int a=100000L, int b=-200000L.

FLOATING POPINT TYPES: The floating point type can hold numbers containing fractional part such as 27.59 etc.

There are two kinds of floating points storage in Java.

- Float
- Double

1.**Float**:

a. The float type values are single precision number.
b. Its minimum value 3.4e-038 to maximum value is 1.7e+0.38.
c. The size of float 4 bytes.
d. The default value is 0.0.
e. This data type is never used for values such as currency.

Example: float f1=324.523f.

2.**DOUBLE**:

- The double types values are double precision number.
- Its minimum value is 3.4e-0.38 to maximum value is 1.7e+308.
- The default value is 0.00.
- The size of double is 8 bytes.
- It will be used for values such as currency.

Example: double d1=123.4567.

3.**CHARACTER**: In order to store character constants in memory, Java provides a character data type called char.

- The char type assumes a size of 2 bytes, but basically it can hold only a single character.
- The minimum value is 0 and maximum value is 65535.

Example: char sex='M'.

4.**BOOLEAN TYPE**:

- Boolean type is used when we want to test a particular condition during the execution of the program. There are only two values that a Boolean type can take: true or false.
- It is denoted by the keyword Boolean.
- It uses only 1 bit of storage.
- It is used in selection and iteration statements.

Example: Boolean flag=true.

03. Explain class and object with example [2019]

Ans.

CLASSES: A class is a user defined blue print or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type . In general, class declarations can include these components, in order:

1. MODIFIERS: A class can be public or has default access.
2. CLASS KEYWORD: Class keyword is used to create a class.
3. CLASS NAME: The name should begin with an initial letter.
4. SUPER CLASS: The name of the class parent, if any, preceded by the keyword elements extends. A class can only extend (sub class) one parent.
5. INTERFACE: A comma-separated list of interfaces by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. BODY: The class body surrounded by braces {}.

Fields are variables that provides the state of the class and its objects.

Methods are used to implement the behaviour of the class and its object.

SYNTAX: class classname[extends superclassname]

{

[fields declaration;]

[methods declaration;]

}

OBJECT: It is a basic unit of object-oriented programming and represents the real life entities. An object consists of:

1. STATE: It is represented by attributes of an object. It also reflects the properties of an object.
2. BEHAVIOUR: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. IDENTITY: It gives a unique name to an object and enables to interact with other objects.

EXAMPLE: Program to implement CLASS and OBJECT.

```
import java.io.*;
class Rectangle
{
    Private int l,b;
    Public void setDimension(int x, int y) {
            l=x; b=y;
    }
```

```
        Public int area (){
                Return l*b;
        }
        Public void display(){
                System.out.println("Length="+l);
            System.out.println("Breadth="+b);
        }
        Public static void main(String ac[]throws Exception{
             BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
            System.out.println("enter length and breadth");
            Int l=Integer.parseInt(br.readline());
            Int b=Integer.parseInt(br.readline());
            Rectangle r=new Rectangle();
            r.setDimension(l,b);
            r.display();
            System.out.println("Area="+r.area());
        }
    }
```

04. Write a java program to illustrate the use of interface [2019]

Ans.

```
import java.io.*;
Interface Cal
{
Void add(int a, int b);
Void sub(int a, int b);
Void mul(int a, int b);
Void div(int a, int b);
}
Public class InterfaceExample
Implements Cal
{
Public void add(int a, int b)
{
Int c=0;
C=a+b;
System.out.println(Addition of "+a+" & "+b+" is: "+c);
}
Public void sub(int a, int b)
{
Int c=0;
C=a-b;
System.out.println("Subtraction of "+a+" & "+b+" is: "+c);
}
Public void mul(int a, int b)
{
```

```
Int c=0;
C=a*b;
System.out.println("Multiplication of "+a+" & "+b+" is: "+c)
}
Public void div(int a, int b)
{
Int c=0; c=a/b;
System.out.println("Division of "+a+" & "+b+" is: "+c)
}
Public static void main(String…args)throws Exception
{
InterfaceExample obj=new InterfaceExample();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter a and b values");
Int a=Integer.parseInt(br.readLine());
Int b=Integer.parseInt(br.readLine());
System.out.println("enter your choice\n 1.Addition\t\t 2.Subtraction\n 3.Multiplication\t
4.Division");
int choice=Integer.parseInt(br.readLine());
switch(choice)
{
Case 1: obj.add(a,b);
Break;
Case 2: obj.sub(a,b);
Break;
Case 3: obj.mul(a,b);
Break;
Case 4: obj.div(a,b);
Break;
Default: System.out.println("the input you have entered is wrong");
Break;
}
}
}
```


05. Define try(), catch(), throw(), throws() and finally() [2019]

Ans.

- **TRY**: The main core of exception handling is try and catch.
  - These keywords work together we cant have a try without a catch or a catch without a try.
  - The general form of try and catch method of exception handling is given below
  - SYNTAX:

```
try{
        //do risky thing
    }
```

```
//try to recover
```

- **CATCH**: A catch block consists of the keyword catch followed by a single parameter between parenthesis that identify the type of exception that the block is to deal with.
  - SYNTAX:

```
try{
//The risky code that can throw one or more exceptions
}
Catch(Exception ex) {
//code to handle the exception
}
```

- **THROW**: The throw class used when the programmer wants to throw an exception explicity and wants to handle it using catch block.

SYNTAX:

```
void method (String filename)
{
    if (filename==null){
        Throw new NullPointerException("File name is null");
        }
}
```

- **THROWS**: The throws clause is used when the programmer does not want to handle the exception in the method and throw it out the method.

SYNTAX:

```
Public some voidMethod() throws SomeException
{
Try{
        //code that may throw SomeException e){
        //somr Exception handling code
        Throw e;
    }
}
```

- **FINALLY BLOCK**: The finally keyword is used to create a block of code that follows a try block.
  - A finally block of code always executes whether or notan exception as occurred.
  - Using finally block, allows us to run any clean up type statements that we want to execute no matter what happens in the try block code.

The syntax for finally block is,

```
Try
{
//protected code
}catch(Exception type1 e1)
{
//catch block
}catch(Exception type2 e2)
{
```

```
//catch block
}finally
{
//The finally block always excutes
}
```

06. Briefly explain usage of adapter class in event handling process.

Ans.

java adapter classes provide the default implementation of listener interfaces. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it saves code.

The adapter classes are found in **java.awt.event**, **java.awt.dnd** and **javax.swing.event** packages. The Adapter classes with their corresponding listener interfaces are given below.

Java.awt.event adapter classes

Adapter class	Listener interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

Java.awt.dnd adapter classes

Adapter class	Listener interface
DragSourceAdapter	DragSourceListener
DragTargetAdapter	DragTargetListener

Javax.swing.event adapter classes

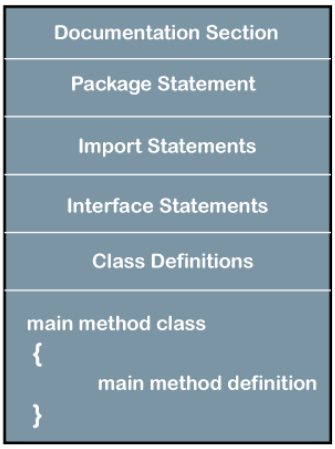
Adapter class	Listener interface
MouseInputAdapter	MouseInputListener
InternalFrameAdapter	InternalFrameListener

07. Explain different java features [2018,2017]

Ans.

1. **Simple**: Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.
2. **Object-Oriented**: In java, everything is an object which has some data and behaviour. Java can be easily extended as it is based on Object Model. Following are some basic concept of OOP's.

a. Object
b. Class
c. Inheritance
d. Polymorphism
e. Abstraction
f. Encapsulation



Structure of Java Program

3. **Robust**: Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.
4. **Platform Independent**:

   Unlike other programming languages such as C, C++ etc which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

   On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security. Any machine with Java Runtime Environment can run Java Programs.
5. **Secure**:

   When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.
6. **Multi-Threading:** Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.
7. **Portable:** Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types.
8. **High Performance:** Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

08. Explain the basic structure of java program with example [2018]

Ans.

Java is an object-oriented programming, **platform independent,** and **secure** programming language that makes it popular. Using the Java programming language, we can develop a wide variety of applications. So, before diving in depth, it is necessary to understand the **basic structure of Java program** in detail

A typical structure of a Java program contains the following elements:

- Documentation Section
- Package Declaration
- Import Statements
- Interface Section
- Class Definition
- Class Variables and Variables
- Main Method Class

- Methods and Behaviours

- **Documentation Section:** The documentation section is an important section but optional for a Java program. It includes **basic information** about a Java program. The information includes the **author's name, date of creation, version, program name, company name,** and **description** of the program.

- **Package Declaration:** The package declaration is optional. It is placed just after the documentation section. In this section, we declare the **package name** in which the class is placed. Note that there can be **only one package** statement in a Java program. It must be defined before any class and interface declaration.

- **Class Definition:** In this section, we define the class. It is **vital** part of a Java program. Without the class, we cannot create any Java program. A Java program may conation more than one class definition. We use the **class** keyword to define the class. The class is a blueprint of a Java program. It contains information about user-defined methods, variables, and constants. Every Java program has at least one class that contains the main () method.

- **Class Variables And Constants:** In this section, we define variables and **constants** that are to be used later in the program. In a Java program, the variables and constants are defined just after the class definition. The variables and constants store values of the parameters. It is used during the execution of the program. We can also decide and define the scope of variables by using the modifiers. It defines the life of the variables.

- **Main Method Class:** In this section, we define the **main () method.** It is essential for all Java programs. Because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class. It must be inside the class. Inside the main method, we create objects and call the methods.

- **Methods And Behaviour:** In this section, we define the functionality of the program by using the methods. The methods are the set of instructions that we want to perform

EXAMPLE:

```
//Java Program to Find Area Of Circle
Public class AreaOfCircle {
  Public static void main(String…args)
{
Int radius=5;
Double area=Math.PI*radius*radius;
System.out.Println("Area of a Circle is"+area);
}
}
```

09. Write program that illustrate the concept of class & objects [2018,2017]

Ans.

```
import java.io.*;
```

```
class Rectangle
{
        private int l,b;
        public void setDimension(int x,int y)
        {
                l=x;b=y;
        }
        public int area(){
                return l*b;
        }
        public void display(){
                System.out.println("Length="+l);
                System.out.println("Breadth="+b);
        }
        public static void main(String ac[])throws Exception
        {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter length and breadth");
        int l=Integer.parseInt(br.readLine());
        int b=Integer.parseInt(br.readLine());
        Rectangle r=new Rectangle();
        r.setDimension(l,b);
        r.display();
        System.out.println("Area="+r.area());
        }
}
```

10. Define array. How to create an array in 1D-array and 2D-array in JAVA [2018]

Ans.

Array is a Collection Of homogenous elements with same name and same type.

<u>One Dimensional Array:</u>

One Dimensional Array in java is always used with only one subscript ([]). A **one-dimensional array** behaves likes a list of variables. You can access the variables of an array by using an index in square brackets preceded by the name of that array. Index value should be an integer.

Datatype [] arrayName; **Or** datatype array Name[]; **Or** datatype arrayName[];

- **datatype** can be a **primitive data type** (int, char, Double, byte etc.)or **Non-primitive data** (Objects).
- **Array Name** is the name of an array
- **[ ]** is called subscript.

<u>Two-Dimensional Array</u>

Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

Two dimensional array: int[ ][ ] twoD_arr = new int[10][20];

11. Write a java program to create a thread implementing the 'Runnable' interface [2018]

Ans.

```
class Th1 extends Thread
{
        public void run ()
        {
                try
                {
                        for (int i=1; i<=10; i++)
                        {
                                System.out.println ("VALUE OF I = "+i);
                                Thread.sleep (1000);
                        }
                }
                catch (InterruptedException ie)
                {
                        System.out.println (ie);
                }
        }
};
class ThDemo1
{
        public static void main (String [] args)
        {
                Th1 t1=new Th1 ();
                System.out.println ("IS T1 ALIVE BEFORE START = "+t1.isAlive ());
                t1.start ();
                System.out.println ("IS T1 ALIVE AFTER START = "+t1.isAlive ());
        }
}
```

12. Discuss with different stages on life cycle of an applet [2018,2017]

Ans.

When an applet is executed within the web browser or in an applet window, it goes through the four stages of its life cycle: initialized, started, stopped and destroyed. These stages correspond to the applet methods init(), start(), stop() and destroy() respectively. All these methods defined in the Applet class which is called automatically by the browser or the applet viewer controlling the applet. All these methods have hollow bodies by default. To perform specific functions, these methods need to be overridden in the user's applet so that the browser can call your code correctly.

We shall now briefly discuss about these methods:

- **Public void init()**:This method is used to perform any initialization that needed for the applet. It is called automatically when the applet is first loaded into the browser and is called only once during the life cycle of the applet. In this method, we generally perform startup activities such as adding GUI components; loading resources such as images, audio, font; creating threads; and getting string parameter values from the APPLET tag in the HTML page.
- **Public void start()**:After the applet is loaded and initialized, the Java environment automatically calls the start() method. It also called when the user returns to the HTML page that contains the applet after browsing through other webpages. This method is used to start the processing for the

applet. For example, Action performed here might include starting an animation or starting other threads of execution. Unlike the init() method, the start() method may be called more than once during the life cycle of an applet.

- **Public void stop():** This method is called automatically by the browser when the user moves off the HTML page containing the applet. It is generally used to suspend the applet's execution so that it does not take up system resources when the user is not viewing the HTML page or has quit the browser. For example Processor intensive activities such as animation, playing audio files or performing calculations in a thread can be stopped which not visible to the user until the user returns to the page.
- **Public void destroy()**:This method called after the stop() method when the user exits the web browser normally. This method is used to clean up resources allocated to the applet that is managed by the local operating system. Like the init() method, it is called only once in the lifetime of the applet.

13. Explain with their general form the different types of if statements a available in java. [2017]

Ans.

The four types of control statements that you can use in java programs based on the requirements are: -

a) if statement
b) nested if statement
c) if-else statement
d) if-else-if statement

1)If statement:

If statement consists a condition, followed by statement or a set of statements as shown below:

```
if(condition){
  Statement(s);
}
```

The statements gets executed only when the given condition is true. If the condition is false then the statements inside if statement body are completely ignored.

Example of if statement

```
public class IfStatementExample {
   public static void main(String args[]){
      int num=70;
      if( num < 100 ){
           System.out.println("number is less than 100");
      }
   }
}
```

2) nested if statement: When there is an if statement inside another if statement then it is called the **nested if statement**.

The structure of nested if looks like this:

```
if(condition_1) {
```

```
  Statement1(s);
  if(condition_2) {
    Statement2(s);
  }
}
```

Statement1 would execute if the condition_1 is true. Statement2 would only execute if both the conditions(condition_1 and condition_2) are true.

Example of Nested if statement

```
public class NestedIfExample {
  public static void main(String args[]){
      int num=70;
       if( num < 100 ){
        System.out.println("number is less than 100");
        if(num > 50){
            System.out.println("number is greater than 50");
          }
       }
  }
}
```

3)If-else statement: This is how an if-else statement looks:

```
if(condition) {
  Statement(s);
}
else {
  Statement(s);
}
```

The statements inside "if" would execute if the condition is true, and the statements inside "else" would execute if the condition is false.

Example of if-else statement

```
public class IfElseExample {
 public static void main(String args[]){
    int num=120;
    if( num < 50 ){
        System.out.println("num is less than 50");
    }
    else {
        System.out.println("num is greater than or equal 50");
    }
  }
}
```

4) **if-else-if Statement:** if-else-if statement is used when we need to check multiple conditions. In this statement we have only one "if" and one "else", however we can have multiple "else if". It is also known as if else if ladder. This is how it looks:

```
if(condition_1) {
  /*if condition_1 is true execute this*/
```

```
  statement(s);
}
else if(condition_2) {
  /* execute this if condition_1 is not met and
   * condition_2 is met
   */
  statement(s);
}
else if(condition_3) {
  /* execute this if condition_1 & condition_2 are
   * not met and condition_3 is met
   */
  statement(s);
}
.
.
.
else {
  /* if none of the condition is true
   * then these statements gets executed
   */
  statement(s);
}
```

Example of if-else-if

```
public class IfElseIfExample {

  public static void main(String args[]){
        int num=1234;
        if(num <100 && num>=1) {
          System.out.println("Its a two digit number");
        }
        else if(num <1000 && num>=100) {
          System.out.println("Its a three digit number");
        }
        else if(num <10000 && num>=1000) {
          System.out.println("Its a four digit number");
        }
        else if(num <100000 && num>=10000) {
          System.out.println("Its a five digit number");
        }
        else {
          System.out.println("number is not between 1 & 99999");
        }
  }
}
```

14. What are the different types of inheritance? Explain any two with example [2017]

Ans.

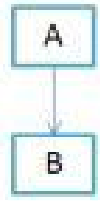
Mainly there are five types of Inheritance,they are:

- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Multiple Inheritance
- Hybrid Inheritance

We will see each one of them one by one with the help of examples and flow diagrams.

1) Single Inheritance

When a class extends another one class only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be a **child class** of A.



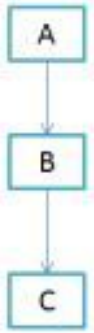
(a) Single Inheritance

Single Inheritance example program in Java

```
Class A
{
  public void methodA()
  {
    System.out.println("Base class method");
  }
}
Class B extends A
{
  public void methodB()
  {
    System.out.println("Child class method");
  }
  public static void main(String args[])
  {
    B obj = new B();
    obj.methodA(); //calling super class method
    obj.methodB(); //calling local method
  }
}
```

2) Multilevel Inheritance

Multilevel inheritance refers to a mechanism where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A.



(d) Multilevel Inheritance

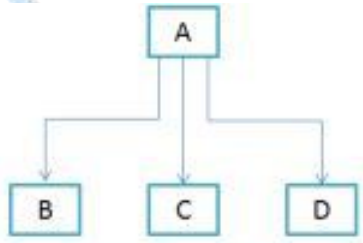
Multilevel Inheritance example program in Java

```
Class X
{
  public void methodX()
```

```
  {
    System.out.println("Class X method");
  }
}
Class Y extends X
{
public void methodY()
{
System.out.println("class Y method");
}
}
Class Z extends Y
{
  public void methodZ()
  {
    System.out.println("class Z method");
  }
  public static void main(String args[])
  {
    Z obj = new Z();
    obj.methodX(); //calling grand parent class method
    obj.methodY(); //calling parent class method
    obj.methodZ(); //calling local method
 }
}
```

3) Hierarchical Inheritance

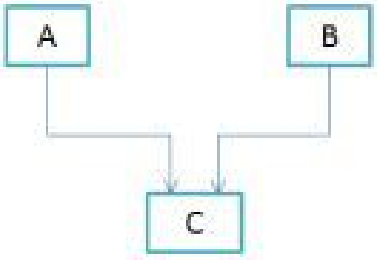
In this inheritance one class is inherited by many **sub classes**. In below example class B, C and D **inherits** the same class A. A is **parent class (or base class)** of B, C & D.



(c) Hierarchical Inheritance

4) Multiple Inheritance

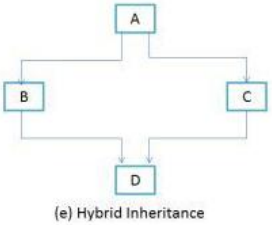
"Multiple Inheritance" refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base classes.



(b) Multiple Inheritance

5) Hybrid Inheritance

In simple terms you can say that Hybrid inheritance is a combination of **Single** and **Multiple inheritance.**



(e) Hybrid Inheritance

15. What is an array? Write program in java to print the numbers in ascending order using one dimensional array [2017]

Ans.

An Array is a Collection of Homogenous Element with same name and type.
For instance we can define an array name salary to represent a set of salaries of a group of Employees. A Particular value is indicated by writting a number called index number or subscript in brackets after the array name.
EX: Salary[10]
Represents the salary of the 10^{th} employee while the complete set of values is referred to as an array. The individual values are called elements. Array can be any variable type. The ability to use a single name to represent a collection of items by specifying the item number enables us to develop concise and efficient program.
For ex: A loop with subscript as the control variable type can be used to read the entire array perform calculations and print out the result.

Program to print the number in Ascending order

```
public class SortAsc {
   public static void main(String[] args) {

      //Initialize array
    int [] arr = new int [] {5, 2, 8, 7, 1};
     int temp = 0;

    //Displaying elements of original array
    System.out.println("Elements of original array: ");
      for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
    }

     //Sort the array in ascending order
    for (int i = 0; i < arr.length; i++) {
        for (int j = i+1; j < arr.length; j++) {
          if(arr[i] > arr[j]) {
              temp = arr[i];
               arr[i] = arr[j];
               arr[j] = temp;
            }
        }
    }

    System.out.println();

    //Displaying elements of array after sorting
    System.out.println("Elements of array sorted in ascending order: ");
    for (int i = 0; i < arr.length; i++) {
       System.out.print(arr[i] + " ");
    }
}
```

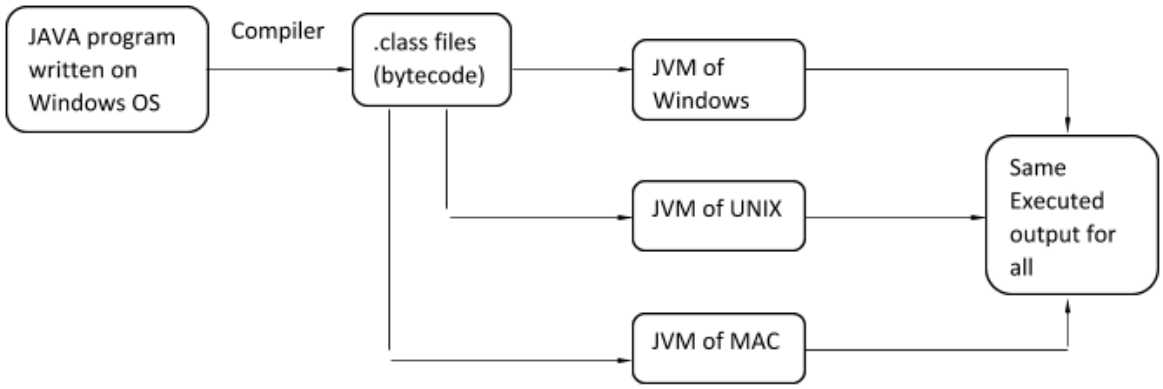
}

16. How java is platform independent. Justify [2016]

Ans.

The Compiling and running of the program process is completely different in JAVA language compared to C/C++ language. Here Bytecodes are produced by an intermediate virtual machine. When the program is compiled by the compiler the intermediate code generated are called Byte codes. Like native codes Byte codes are not executable codes. The virtual machine present in the machine executes these byte codes. These bytecodes can be executed in any other platform irrespective of their generated platform. When a program is written and compiled in JAVA a separate file is created for a compiled program. This file (.class) is called bytecode in java. The .class file created cannot be executed directly. It does not include executable codes. Instead, it will be converted into executable code by a virtual machine in the system. These bytecodes generated by the compiled program are to achieve the purpose of platform independency. Byte code generated in a particular platform can be executed in any other platform i.e., the byte code generated in windows OS can also be executed in Unix OS. The one which makes this possible is the JVM (JAVA VIRTUAL MACHINE). When the program is written and compiled, the compiler sends the generated bytecodes to the JVM resent in the machine and this JVM converts the byte codes into native code which is readable by a particular machine. Thus, the output is displayed. Irrespective of the platform the JVM belongs to, the generated bytecode can run on any JVM. The outputs of the bytecode run on any JVM will be the same. Hence the JAVA is called platform independent language.

Diagram



17. Explain any 2 branching statements in java [2016]

Ans.

Branching statements are the statements used to jump the flow of execution from one part of a program to another. The branching statements are mostly used inside the control statements. Java has mainly three branching statements, i.e., continue, break, and return. The branching statements allow us to exit from a control statement when a certain condition meet. In java **continue** and break statements are two essential branching statements used with the control statements.

Continue Statement

The continue statement skips the current execution and pass the control to the start of the loop. The continue statement is another branching statement used to immediately jump to the next iteration of the loop. It is a special type of loop which breaks current iteration when

the condition is met and start the loop with the next iteration. In simple words, it continues the current flow of the program and stop executing the remaining code at the specified condition.

When we have a nested for loop, and we use the continue statement in the innermost loop, it continues only the innermost loop. We can use the continue statement for any control flow statements like for, while, and do while.

Syntax

```
control-flow-statement;
continue;
```

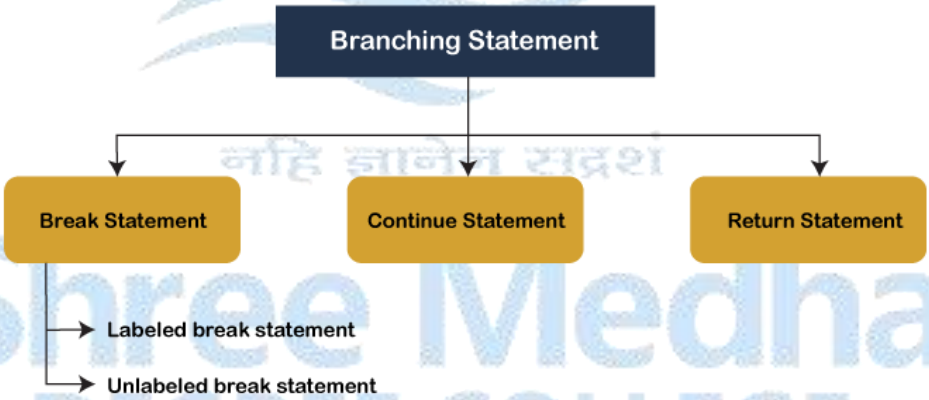
Break Statement

The break statement breaks or terminates the loop and transfers the control outside the loop. The unlabelled break statement is used to terminate the loop that is inside the loop. It is also used to stop the working of the switch statement. We use the unlabelled break statement to terminate all the loops available in Java.

Syntax

```
    for (int; testExpression; update) {
  //Code
  if(condition to break){
     break;
    }
  }
```

Diagram



18. Write a java program to illustrate method overloading [2016]

Ans.

```
class Figure
 {
double dim1;
double dim2;
Figure(double a, double b)
 {
dim1 = a;
 dim2 = b;
}
double area()
{
System.out.println("Area for Figure is undefined.");
```

```
return 0;
 }
 }
 class Rectangle extends Figure
 {
Rectangle(double a, double b)
{
 super(a, b);
}
 // override area for rectangle
 double area()
{
 System.out.println("Inside Area for Rectangle.");
   return dim1 * dim2;
}
}
 class Triangle extends Figure
{
          Triangle(double a, double b)
{
 super(a, b);
          }
// override area for right triangle
double area()
 {
                    System.out.println("Inside Area for Triangle.");
 return dim1 * dim2 / 2;
}
 }
class FindAreas
{
 public static void main(String args[])
{
Figure f = new Figure(10, 10);
 Rectangle r = new Rectangle(9, 5);
Triangle t = new Triangle(10, 8);
 Figure figref;
figref = r;
 System.out.println("Area is " + figref.area());
 figref = t;
 System.out.println("Area is " + figref.area());
figref = f;
 System.out.println("Area is " + figref.area());
}
}
```

19. Explain applet skeleton with an example. **[2016]**

Ans.

```
// An Applet skeleton.
```

```
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet {
        // Called first.
        public void init() {
                // initialization
        }
        /* Called second, after init().
        Also called whenever
        the applet is restarted. */
        public void start() {
                // start or resume execution
        }
        // Called when the applet is stopped.
        public void stop() {
                // suspends execution
        }
        /* Called when applet is terminated.
        This is the last
        method executed. */
        public void destroy() {
                // perform shutdown activities
        }
        // Called when an applet's window must be restored.
        public void paint(Graphics g) {
                // redisplay contents of window
        }
}
```

20. Write a note on exception handling [2016]

Ans.

Exception handling is one of the most important feature of java programming that allows us to handle the runtime errors caused by exceptions. An Exception is an unwanted event that interrupts the normal flow of the program. When an exception occurs program execution gets terminated. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled in Java. By handling the exceptions we can provide a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user. If an exception occurs, which has not been handled by programmer then program execution gets terminated and a system generated error message is shown to the user.

Advantage of exception handling

Exception handling ensures that the flow of the program doesn't break when an exception occurs. For example, if a program has bunch of statements and an exception occurs midway after executing certain statements then the statements after the exception will not execute and the program will terminate abruptly.

By handling we make sure that all the statements execute and the flow of program doesn't break.

21. Explain different sources of event. [2016]

Ans.

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Button
public void addActionListener(ActionListener a){}
MenuItem
public void addActionListener(ActionListener a){}
TextField
public void addActionListener(ActionListener a){}
public void addTextListener(TextListener a){}
TextArea
public void addTextListener(TextListener a){}
Checkbox
public void addItemListener(ItemListener a){}
Choice
public void addItemListener(ItemListener a){}
List
public void addActionListener(ActionListener a){}
public void addItemListener(ItemListener a){}

10M QUESTION & ANSWER

01. Explain in detail the features of Java programming [2019]

Ans.

Java is a simple, general-purpose, object-oriented, distributed, Compiled and interpreted, robust, secure, portable, multithreaded and dynamic programming language developed by **James Gosling** and his team at sun Microsystem of USA in the year 1991.

The features or characteristics of java are:

- Object oriented.
- Secured.
- Robust
- Complied and interpreted.
- Distributed
- Portable
- Multithreaded.
- Dynamic and extensible.
- High performance.
- Ease of development.
- Simple, small and familiar.

Object oriented:

i. Java is a true object oriented language.
ii. Almost everything in java is an object.
iii. All program code and data reside within objects and classes.
iv. The object model in java is simple and easy to extend.

Secure:

i. Security becomes an important issue for a language that is used for programming on internet.
ii. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet.
iii. The absence of pointers in java ensure that programs cannot gain access to memory locations without proper authorization

Robust:

i. java is a robust language.
ii. It provides many safeguards to ensure reliable code.
iii. It is designed as a garbage collected language relieving the programmers virtually all memory management problems.
iv. It also incorporates the concept of exception handling which capture serious errors and eliminates any risk of crashing the system.

Compiled and interpreted:

i. usually a computer language is either compiled or interpreted.
ii. Java compiler translates source code into what is know as bytecode instruction. Bytecode are not machine instruction.
iii. Java interpreter generates machine code that can be directly executed by the machine that is running the java program.

Distributed:

i. Java is designed as a distributed language for creating applications on networks.
ii. It has the ability to share both data and programs.
iii. Java applications can open and access remote objects on internet as easily as they can do in a local system.

Portable:

i. The most significant contribution of java over other languages is its portability.
ii. Java ensures portability in two ways. First, java compiler generates bytecode instructions that can be implemented on any machine. secondly, the size of the primitive data type is machine independent.
iii. Java program can be easily moved from one computer system to another, anywhere and anytime.

Multithreaded:

i. Multithreaded means handling multiple tasks simultaneously java supports multithreaded program.
ii. This means that we need not wait for the application to finish one task before beginning another.
iii. Example: we can listen to an audio clip while scrolling a page and at the same time download an applet from a distant computer.

Dynamic and extensible:

i.Java is a dynamic languages.java is capable of dynamically linking in new class libraries, methods and objects.
ii.Java program support functions written in other languages such as C and C++.These functions are known as native methods.
iii.This facility enables the programmers to use he efficient functions available in these languages. Native methods are linked dynamically at runtime.

High performance:

i. Java performance is impressive for interpreted languages mainly due to the use of intermediate bytecode.
ii.Java architecture is also designed to reduce overheads during runtime.
iii.The incorporation of multithreading enhances the overall executive speed of java programs.

simple ,small and familiar:

i.java is a small and simple language.
ii.Java code "looks like a c++" code.
iii.Familiarity is another striking features of java.to make the languages look familiar to the existing programmers.it was modeled on C and C++ languages.

02. What is Looping? Explain different types of Looping [2019]

Ans.

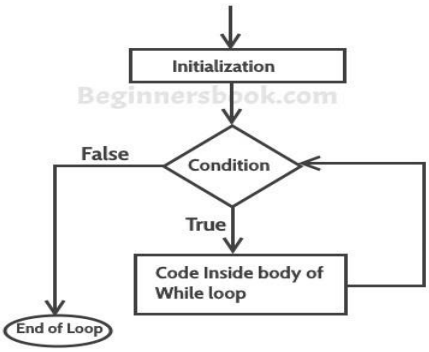
The process of repeatedly executing a single or collection of statements until the given condition is FALSE is called looping or iterative or repetitive statements.

There are 3 types of iterative statements:

i) while loop.
ii) Do-while loop
iii) For loop

- **While-loop:** The while loop is used to repeat a block of statements for given number of times, until the given condition is false.
  - if the condition is **true** then statements inside the loop will be exceuted. if the condition is **false** then it will come out of loop
  - it is also called as entry control or pre-check loop
  - **syntax:**

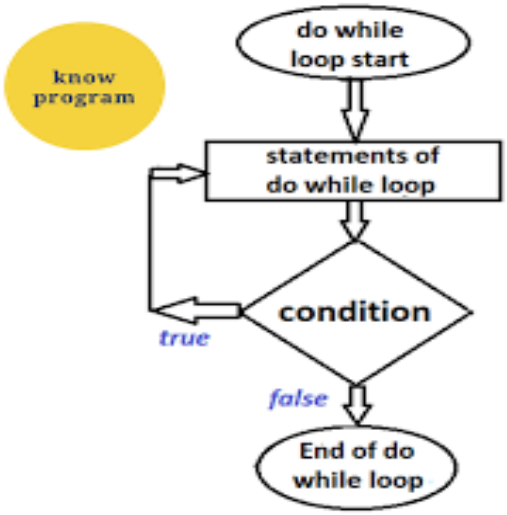
```
while(condition)
{
//statements
}
```



Example:

```
int i=1;
System.out.println("list of 1 to 10 numbers");
While(i<=10)
{
System.out.println(i);
i++;
}
```

➢ **do-while loop:** this loop is similar to the while loop, it will test the condition at the end of the loop, this means the statements inside the loop will execute at least once, even if the condition is FALSE. It is also called exit control or post check loop.



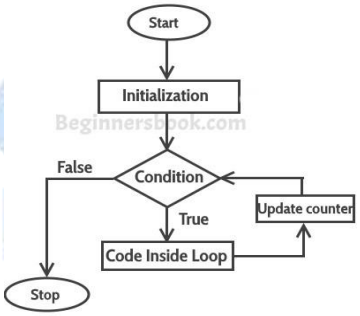
Syntax:

```
do
{
//statements;
} while(condition);
```

Example:

```
row=1;
do
{
Column=1;
do
{
Y=row*column;
System.out.println(" "+y);
Column=column+1;
}
While(column<=3)
System.out.println("\n");
row=row+1;
}
While(row<=3);
```


➢ **For loop:** it is used to repeat a block of statements for a given number of times, until the given condition is False.



Syntax:

```
For (initialization; test condition; increment/decrement operator)
{
//statements;
}
```

Example:

```
sum=0;
for (n=1; n<=10; n++)
{
sum=sum+n*n;
}
```

03. List out different drawing method of a graphics class. Explain each with an example [2019]

Ans.

The Graphics class is the abstract super class for all graphics contexts which allow an application to draw onto components that can be realized on various devices, or onto off-screen images as well.

A Graphics object encapsulates all state information required for the basic rendering operations that Java supports. State information includes the following properties.

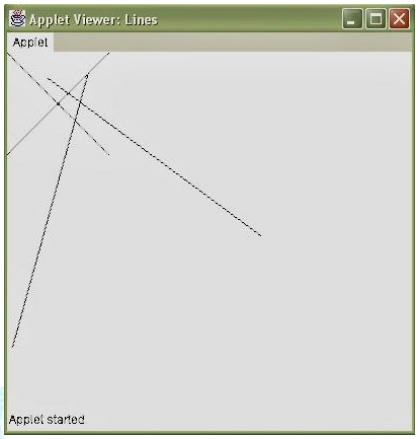
- The Component object on which to draw.
- A translation origin for rendering and clipping coordinates.
- The current clip.
- The current color.
- The current font.
- The current logical pixel operation function.
- The current XOR alternation color

Displaying Graphics in Applet
Java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods for Graphics class:

- **public abstract void drawString(String str, int x, int y)**: is used to draw the specified string.
- **public void drawRect(int x, int y, int width, int height)**: draws a rectangle with the specified width and height.
- **public abstract void fillRect(int x, int y, int width, int height)**: is used to fill rectangle with the default color and specified width and height.
- **public abstract void drawOval(int x, int y, int width, int height)**: is used to draw oval with the specified width and height.
- **public abstract void fillOval(int x, int y, int width, int height)**: is used to fill oval with the default color and specified width and height.
- **public abstract void drawLine(int x1, int y1, int x2, int y2)**: is used to draw line between the points(x1, y1) and (x2, y2).
- **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)**: is used draw the specified image.
- **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)**: is used draw a circular or elliptical arc.
- **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)**: is used to fill a circular or elliptical arc.
- **public abstract void setColor(Color c)**: is used to set the graphics current color to the specified color.
- **public abstract void setFont(Font font)**: is used to set the graphics current font to the specified font.

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Lines" width=300 Height=250>
</applet>
*/
public class Lines extends Applet
{
      public void paint(Graphics g)
      {
            g.drawLine(0,0,100,100);
            g.drawLine(0,100,100,0);
            g.drawLine(40,25,250,180);
            g.drawLine(5,290,80,19);
      }
}
```



04. Explain applet context and show document with an example [2019,2016]

Ans.

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.

Method	Description
Applet getApplet(String *appletName*)	Returns the applet specified by *appletName* if it is within the current applet context. Otherwise, **null** is returned.
Enumeration<Applet> getApplets( )	Returns an enumeration that contains all of the applets within the current applet context.
AudioClip getAudioClip(URL *url*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url*.
Image getImage(URL *url*)	Returns an **Image** object that encapsulates the image found at the location specified by *url*.
InputStream getStream(String *key*)	Returns the stream linked to *key*. Keys are linked to streams by using the **setStream( )** method. A **null** reference is returned if no stream is linked to *key*.
Iterator<String> getStreamKeys( )	Returns an iterator for the keys associated with the invoking object. The keys are linked to streams. See **getStream( )** and **setStream( )**.
void setStream(String *key*, InputStream *strm*) throws IOException	Links the stream specified by *strm* to the key passed in *key*. The *key* is deleted from the invoking object if *strm* is **null**.
void showDocument(URL *url*)	Brings the document at the **URL** specified by *url* into view. This method may not be supported by applet viewers.

One application of Java is to use active images and animation to provide a graphical means of navigating the Web that is more interesting than simple text-based links. To allow your

applet to transfer control to another URL, you must use the **showDocument**( ) method defined by the **AppletContext** interface. **AppletContext** is an interface that lets you

get information from the applet's execution environment. The methods defined by **AppletContext** are shown in Table 23-2. The context of the currently executing applet is obtained by a call to the **getAppletContext**( ) method defined by Applet.

05. Define event. Explain delegation event model with neat diagram [2019]

Ans.

The GUI in Java processes the interactions with users via mouse, keyboard and various user controls such as button, checkbox, text field, etc. as the events. These events are to be handled properly to implement Java as an Event-Driven Programming.

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event

The events can be broadly classified into two categories:

- **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.

- **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Delegation Event Model

Back in the old days, Java used a Chain of Responsibility pattern to process events. For example, when a button is clicked, a event is generated, which then is passed through a chain of components. The chain of components is defined by the hierarchy of classes and interfaces. An event is caught and handled by the handler class. This mechanism was used by Java version 1.0, which is very different from the event handling scheme of Java version 1.1 onwards. Old methods are still supported, but deprecated and hence not recommended for new programs. A modern approach is based on the delegation event model.

The processing model of Java 1.1's event hierarchy facilitates more than one receiver subscription. The subscriber thus can send notifications to all of them in response to an change or updates. This mechanism reminds one of the Observer Pattern. In the delegation event model, a class designated as an event source generates an event and sends it to one or more listeners. The responsibility of handling the event process is handed over to its listeners. The listeners classes wait in the vicinity, to spring into action only when its is

poked by the event that it is interested in. The design scheme is neatly decoupled from the main application logic that generates the event. However, the listeners must register or agree with the event source class to receive any notification. This means that a particular event is processed only by a specific listener. The overhead of going through a chain of containment hierarchy of Java 1.0 is eliminated. Java 1.0 used to make events go through many listeners that do not process the particular event, wasting valuable time. The modern approach made the delegation simple, efficient, and effective in view of its decoupled nature and performance issues.

06. Explain the basic structure of java programming [2017,2016]

Ans.

Java program may contain many classes of which only one class defines a main method.

Structure of Java Program

- Classes contain data members and methods that operate on the data members of the class.
  Methods may contain data type declaration and executable statement.

Documentation section: it consists of set of comment lines which are non-executable statements that gives the name of the program and other details.

- Comments must explain why and what of classes and how of algorithms.

There are two types of comments line:

i) Single comment line: start with //
ii) Multi-comment line: start with /* and end with */

package statement: The first statement allowed in java file is a **package** statement. It declares a package name and informs the compiler that the classes defined belong to this package.it is optional

example: package statement;

import statement: the next thing after a package statement (but before any class definition) may be a number of import statements. It is similar to #include statement in C.

example: import student. Test;

- this statements instruct the interpreter to load the test class contained in the student package.
- Using these students, we can have access to classes that are part of other named packages.

Interface statements: An interface is like a class but includes a group of method declarations. It is also optional.

- it is used only when we implement the multiple inheritance feature in the program.

Class definitions: A java program may contain multiple class definitions.

- Classes are the primary and essential elements of a java programs.
- Classes are used to map the objects of real world problems.
- The number of classes used depends on the complexity of problem.

Main method class: every java standalone program requires a main method as its starting point, this class is the essential part of a java program.

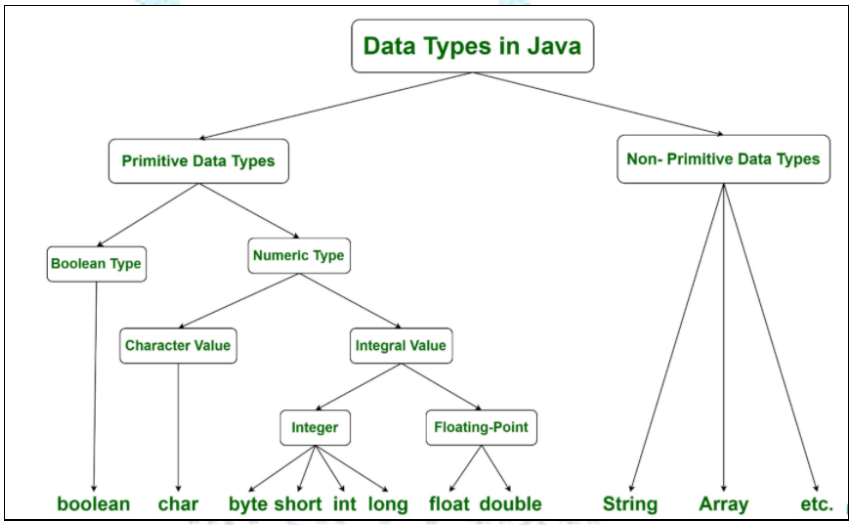
the main method creates objects of various classes and establishes communications between them. On reaching the end of main, the program terminates and the control passes back to the operating system.

07. Define datatypes. Explain different datatype supported by java with example [2018, 2017]

Ans.

Java is **statically typed and also a strongly typed language** because, in Java, each type of data (such as integer, character, hexadecimal, packed decimal, and so forth) is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types.

Data Types in Java:



Java has two categories of data:

- **Primitive Data Type:** such as Boolean, char, int, short, byte, long, float, and double
- **Non-Primitive Data Type or Object Data type:** such as String, Array, etc.

Primitive Data Type:

Primitive data are only single values and have no special capabilities.

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'', '\n', 'β'	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

There are **8 primitive data types**:

- **Boolean:** Boolean data type represents only one bit of information **either true or false**, but the size of the Boolean data type is **virtual machine-dependent**. Values of type Boolean are not converted implicitly or explicitly (with casts) to any other type. But the programmer can easily write conversion code.

  Syntax: boolean booleanVar;

- **Byte:** The byte data type is an 8-bit signed two's complement integer. The byte data type is useful for saving memory in large arrays.

  Syntax: byte byteVar;

- **Short:** The short data type is a 16-bit signed two's complement integer. Similar to byte, use a short to save memory in large arrays, in situations where the memory savings actually matters.

  Syntax: short shortVar;

- **Int**: It is a 32-bit signed two's complement integer.

  Syntax: int intVar;

- **long:** The long data type is a 64-bit two's complement integer.

  Syntax: long longVar;

- **float:** The float data type is a single-precision 32-bit IEEE 754 floating-point. Use a float (instead of double) if you need to save memory in large arrays of floating-point numbers.

  Syntax: float floatVar;

- **double:** The double data type is a double-precision 64-bit IEEE 754 floating-point. For decimal values, this data type is generally the default choice.

  Syntax: double doubleVar;

- **char**: The char data type is a single 16-bit Unicode character.

  Syntax: char charVar;

  Program Example:

```
// Java program to demonstrate
// primitive data types in Java
class Smc {
  public static void main(String args[])
  {
            // declaring character
```

```
        char a = 'G';
        // Integer data type is generally
        // used for numeric values
        int i = 89;
        // use byte and short
        // if memory is a constraint
        byte b = 4;
        // this will give error as number is
        // larger than byte range
        // byte b1 = 7888888955;
        short s = 56;
        // this will give error as number is
        // larger than short range
        // short s1 = 87878787878;
        // by default fraction value
        // is double in java
        double d = 4.355453532;
        // for float use 'f' as suffix
        float f = 4.7333434f;
        System.out.println("char: " + a);
        System.out.println("integer: " + i);
        System.out.println("byte: " + b);
        System.out.println("short: " + s);
        System.out.println("float: " + f);
        System.out.println("double: " + d);
    }
}
```

Non-Primitive Data Type or Reference Data Types:

The **Reference Data Types** will contain a memory address of variable value because the reference types won't store the variable value directly in memory. They are **strings**, **objects**, arrays, etc.

- **String**: Strings are defined as an array of characters. The difference between a character array and a string in Java is, the string is designed to hold a sequence of characters in a single variable whereas, a character array is a collection of separate char type entities.

- Unlike C/C++, Java strings are not terminated with a null character.
  Below is the basic syntax for declaring a string in Java programming language.
  Syntax:
  <String_Type> <string_variable> = "<sequence_of_string>";

  Example:
  // Declare String without using new operator

  String s = "SMDC";
  // Declare String using new operator
  String s1 = new String("SMDC");

➢ **CLASS**: A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

✓ **Modifiers**: A class can be public or has default access.

- ✓ **Class name:** The name should begin with a initial letter (capitalized by convention).
- ✓ **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- ✓ **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- ✓ **Body:** The class body surrounded by braces, { }.

➢ **OBJECT**: It is a basic unit of Object-Oriented Programming and represents the real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

- ✓ **State**: It is represented by attributes of an object. It also reflects the properties of an object.
- ✓ **Behaviour**: It is represented by methods of an object. It also reflects the response of an object with other objects.
- ✓ **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

➢ **INTERFACE:** Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, nobody).

- ✓ Interfaces specify what a class must do and not how. It is the blueprint of the class.
- ✓ An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.
- ✓ If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.
- ✓ A Java library example is, Comparator Interface. If a class implements this interface, then it can be used to sort a collection.

➢ **ARRAY:** An array is a group of like-typed variables that are referred to by a common name. Arrays in Java work differently than they do in C/C++. The following are some important points about Java arrays.

- ✓ In Java, all arrays are dynamically allocated. (discussed below)
  Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using size.
- ✓ A Java array variable can also be declared like other variables with [] after the data type.
- ✓ The variables in the array are ordered and each has an index beginning from 0.
- ✓ Java array can be also be used as a static field, a local variable or a method parameter.
- ✓ The **size** of an array must be specified by an int value and not long or short.
- ✓ The direct superclass of an array type is Object.

08. Write a java program to take data from the board to find greatest of three numbers using conditional operator? [2018,2017]

Ans.

```
import java.util.Scanner;
public class LargestNumberExample1
{
public static void main(String[] args)
{
int a, b, c, largest, temp;
```

```
//object of the Scanner class
Scanner sc = new Scanner(System.in);
//reading input from the user
System.out.println("Enter the first number:");
a = sc.nextInt();
System.out.println("Enter the second number:");
b = sc.nextInt();
System.out.println("Enter the third number:");
c = sc.nextInt();
//comparing a and b and storing the largest number in a temp variable
temp=a>b?a:b;
//comparing the temp variable with c and storing the result in the variable
largest=c>temp?c:temp;
//prints the largest number
System.out.println("The largest number is: "+largest);
}
}
```

OUTPUT:

```
Enter the first number:
23
Enter the second number:
11
Enter the third number:
67
Largest Number is: 67
```

09. Define Package. Explain the steps to implement packages in java [2018,2017]

Ans.

A **java package** is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Here, we will have the detailed learning of creating and using user-defined packages.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

Step 01: Create a package(folder) named According to Naming Conventions(p1).

Step 02: Create a file named MainClass.java inside p1 package.

Step 03: Write the following code in MainClass.java(MyCalculator.java) file.

```
//MyCalculator.java
```

```
package p1;
public class MyCalculator
{
   public int Add(int x,int y)
   {
      return x+y;
   }
   public int Subtract(int x,int y)
   {
      return x-y;
   }
   public int Product(int x,int y)
   {
      return x*y;
   }
}
```

Step 04: Create a file named TestPackage.java outside p1 package.
Step 05: Write the following code in TestPackage.java file.

```
//TestPackage.java
   import p1.MyCalculator;
   class TestPackage
   {
      public static void main(String[] args)
      {

         MyCalculator M = new MyCalculator();

         System.out.print("\n\n\tThe Sum is : " + M.Add(45,10));
         System.out.print("\n\n\tThe Subtract is :" + M.Subtract(45,10));
         System.out.print("\n\n\tThe Product is :" + M.Product(45,10));

      }
   }
```

Step 06: Compile and Run TestPackage.java and get the output.

Output:

```
The Sum is: 55
The Subtract is: 35
The Product is: 450
```

10. Explain the getDocumentBase() and getCodeBase() methods with a program.
Ans.

The **getCodebase()** method is also commonly used to establish a path to other files or folders that are in the same location as the class being run. The **getDocumentBase()** method is used to return the URL of the directory in which the document is resides.

URL getCodeBase(): Gets the base URL.

URL getDocumentBase(): Gets the URL of the document in which the applet is embedded.

In most of the applets, it is required to load text and images explicitly. Java enables loading data from two directories. The first one is the directory which contains the HTML file that started the applet (known as the **document base).** The other one is the directory from which the class file of the applet is loaded (known as the **code base).** These directories can be obtained as URL objects by using getDocumentBase ()and getCodeBase ()methods respectively. You can concatenate these URL objects with the string representing the name of the file that is to be loaded.

Here is the java code for the program GetDocumentBase and getCodeBase Example :

```
/*
   <applet code="GetDocumentBaseExample" width=350 height=250>
   </applet>
   */
  import java.applet.Applet;
   import java.awt.Graphics;
   import java.net.URL;
   import java.awt.*;
   import java.awt.event.*;

   public class GetDocumentBaseExample extends Applet{

         public void paint(Graphics g){
         String message;

    //getCodeBase() method gets the base URL of the directory in which contains this
applet.

       URL appletCodeDir=getCodeBase();
       message = "Code Base : "+appletCodeDir.toString();
       g.drawString(message,10,90);

      // getDocumentBase() Returns an absolute URL of the Document
        URL appletDocDir = getDocumentBase();
       message="Document Base : "+appletDocDir.toString();
       g.drawString(message,10,120);
       g.drawString("https://ecomputernotes.com", 200, 250);
         }
   }
```

11. Explain the key event classes and mouseEvent classes with an example [2018,2017]

Ans.

Mouse Events:

Event Listener Interfaces

- ***java.awt.Event*** package : Listener interfaces to handle events.

All the interfaces and the methods in them.

Event	Generated by	Listener Interface	Methods in them
Action event	Pressing a button	ActionListener	actionPerformed()
Adjustment event	Scroll bar is manipulated	AdjustmentListener	adjustmentValueChanged()
Component event	A component is hidden, moved, resized, or shown	ComponentListener	componentHidden() componentMoved() componentResized() componentShown()
Container event	A component is added or removed from a container	ContainerListener	componentAdded() componentRemoved()
Focus event	A component gains or loses focus	FocusListener	focusGained() focusLost()
Item event	An item is selected or deselected	ItemListener	itemStateChanged()
Key event	A key is pressed, released or typed	KeyListener	keyPressed() keyReleased() keyTyped()
Mouse event	Mouse button is clicked, pressed or released. Mouse pointer enters leaves a component	MouseListener	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
Mouse event	Mouse pointer is dragged or moved	MouseMotionListener	mouseDragged() mouseMoved()
Text event	Text value is changed	TextListener	textValueChanged()
Window event	A window is activated, closed, deactivated, deiconfied, opened, or quit	WindowListener	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()

Getting ready to handle events
Step 1: Import classes from the package:

import java.awt.event.*;

Step 2: Implement the appropriate listener interfaces.

public class sample extends Applet implements listenerinterface1{
} //Several listener interfaces can be listed, separated by commas.

Step 3: Register the listener object with the appropriate event source.

event_source_object.addevent_listenername(event_listener_object);
//Normally, you use the keyword this for event_listener_object.

Step 4: Define *all* the methods of the implemented listener interfaces. I
Mouse Events

Event Class	MouseEvent
Listener Interface	MouseListener
Listener Methods	public void mouseEntered(MouseEvent event) public void mouseExited(MouseEvent event) public void mouseClicked(MouseEvent event) public void mousePressed(MouseEvent event) public void mouseReleased(MouseEvent event) By looking at the name of the methods, it is clear which activity invokes which method. event.getX() returns (int) the x coordinate of mouse position. event.getY() returns (int) the y coordinate of mouse position. event.getClickCount() returns (int) the number of mouse clicks.

Four steps:

Step 1: Include the following import statement**:**
import java.awt.event.*;

Step 2: Include ***Implements MouseListener*** :

public class w2 extends Applet implements MouseListener {
}

Step 3: Register ***addMouseListener*** method in the applet.
this.addMouseListener (this);

Step 4: Define the five *MouseListener* methods:

```
public void mouseClicked(MouseEvent event){
}
public void mouseEntered(MouseEvent event){
}
public void mouseexited(MouseEvent event){
}
public void mousePressed(MouseEvent event){
}
```

```
public void mouseReleased(MouseEvent event){
}
```

Keyboard Events

Whenever a key is pressed, an object of the KeyEvent class is generated.

Event Class	KeyEvent
Listener Interface	KeyListener
Listener Methods	*public void keyPressed(KeyEvent event)* Invoked when a key is pressed. *public void keyReleased(KeyEvent event)* Invoked when a key is released. *public void keyTyped(KeyEvent event)* Invoked when a key is typed (pressed and released).
KeyEvent class	For a list of Class variables (virtual key codes), which are defined as static and final, see Virtual Key Codes table below. *event.getKeyChar()* returns (char) the character pressed. *event.getKeyCode()* returns (int) the "virtual key code", see table below. *event.getModifiers().* event.isShiftDown() returns (boolean) true if shift key is down. *event.isControlDown()* returns (boolean) true if control key is down. *event.isAltDown()* returns (boolean) true if alt key is down *KeyEvent.getKeyModifiersText(int modifiers*) returns (String). *KeyEvent.getKeyText(int keycode)* returns (String).

Virtual Key Codes

- Refer to the key on the keyboard returned by *event.getKeyCode()*
- Usedas class variables.

- Produce a numeric value - but - use the symbol in the program. Nno guarantee that the numeric value produced is the same on all platforms.

Alpha keys	VK_A through VK_Z
Numeric keys	VK_0 through VK_9
Function keys	VK_F1 through VK_F12
Action keys	VK_BACK_SPACE VK_ESCAPE, VK_PRINTSCREEN, VK_SCROLL_LOCK, VK_PAUSE
Num pad number keys	VK_NUMPAD0 through VK_NUMPAD9
Other Num pad keys	VK_NUM_LOCK, VK_SLASH, VK_MULTIPLY, VK_SUBTRACT, VK_ADD, VK_HOME, VK_UP, VK_PAGE_UP, VK_LEFT, VK_RIGHT, VK_END, VK_DOWN, VK_PAGE_DOWN, VK_INSERT, VK_DELETE, VK_ENTER
Modifier keys	VK_ALT, VK_CAPS_LOCK, VK_CONTROL, VK_META, VK_SHIFT
Punctuation keys	VK_SPACE, VK_ENTER, VK_TAB, VK_BACK_QUOTE, VK_SLASH, VK_BACK_SLASH, VK_OPEN_BRACKET, VK_CLOSE_BRACKET, VK_PERIOD, VK_QUOTE, VK_SEMICOLON, VK_SEPARATOR, VK_COMMA, VK_DECIMAL, VK_EQUALS,
Other keys	VK_ACCEPT, VK_CANCEL, VK_CLEAR, VK_CONVERT, VK_FINAL, VK_HELP, VK_KANA, VK_KANJI, VK_MODECHANGE, VK_NONCONVERT, VK_UNDEFINED,

Note
To check for an upper case letter use:
if (event.getKeyCode() == VK_A && KeyEvent.isShiftDown())

Similarly use **event.isControlDown()**and **event.isAltDown()**.
Example
The following program displays several text boxes. Type any key in top box. The other boxes in the applet show information about the key pressed.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Keys1 extends Applet implements KeyListener{
```

```
        TextField input;
        TextField output1, output2, output3, output4, output5;
        TextField output6, output7, output8;
        public void init(){
        input = new TextField(30);
        input.addKeyListener(this);
        output1 = new TextField(30);
        output2 = new TextField(30);
        output3 = new TextField(30);
        output4 = new TextField(30);
        output5 = new TextField(30);
        output6 = new TextField(30);
        output7 = new TextField(30);
        output8 = new TextField(30);
        add(input);
        add(output1);
        add(output2);
        add(output3);
        add(output4);
        add(output5);
        add(output6);
        add(output7);
        add(output8);
        }
public void keyTyped(KeyEvent event){
}
public void keyPressed(KeyEvent event){
        char char1 = event.getKeyChar();
        int n = event.getKeyCode();
        int modifiers = event.getModifiers();
        output1.setText("You pressed "+char1);
        output2.setText("Virtual key code value = "+n);
        output3.setText("Name of key pressed = "+KeyEvent.getKeyText(n));
        output4.setText("Shift was pressed = "+event.isShiftDown());
        output5.setText("Control key down = "+event.isControlDown());
        output6.setText("Alt key was pressed = "+event.isAltDown());
        output7.setText("Modifiers value = "+event.getModifiers());
        output8.setText("Modifier text
        ="+KeyEvent.getKeyModifiersText(modifiers));
}
public void keyReleased(KeyEvent event) {
}
}
```

12. Define inheritance. Explain how we can implement multiple Inheritance in Java [2018]

Ans.

The mechanism of deriving a new class from an old class is called inheritance. The old class is known as base class or super class or parent class and the new one is called the subclass or derived class or child class. It can be defined as the process of acquiring the properties of one object from another object.

- A class that is inherited is called super class.

- A class that does inheriting is called sub class.
- Super class is also called as base class or parent class.
- A sub class is also called as child class or derived class.

The inheritance allows subclasses to inherit all the variables and methods of their parent classes. Inheritances are in different forms:

i. Single Inheritance (only one super class)
ii. Multiple Inheritance (several super classes)
iii. Hierarchical Inheritance (one super class, many subclasses)
iv. Multilevel Inheritance (Derived from a derived class)

Defining a Subclass:

```
class subclassname extends superclassname
{
    variables declaration;
    methods declaration;
}
```

The keyword extends signifies that the properties of the superclassname are extended to the subclassname. The subclass will now contain its own variables and methods as well as those of the superclass. This kind of situation occurs when we want to add some more properties to an existing class without actually modifying it.

- Multiple Inheritance:

Multiple inheritance a feature of some object-oriented programming languages in which a class or an object inherits characteristics and properties from more than one parent class or object. This is contrary to the single inheritance property, which allows an object or class to inherit from one specific object or class.

Example,

```
class ParentClass1
{
    void show()
    {
        System.out.println("ParentClass1");
    }
}
class ParentClass2
{
    void show()
    {
        System.out.println("ParentClass2");
    }
}
class Subclass extends ParentClass1,ParentClass2{
        public static void main(String[] args) {
                SubClass obj=new SubClass();
```

```
            obj.show();
        }
}
```

Object Oriented Programming provides a user the feature of multiple inheritance, wherein a class can inherit the properties of more than a single parent class. In simpler terms, multiple inheritance means a class extending more than one class.
Multiple inheritance in java means one class implementing two or more than two interfaces simultaneously. Multiple inheritance in java means one class accessing the states and behaviour of two or more than two interfaces simultaneously.

- Java does not support multiple inheritance. Since multiple inheritance is an important concept in OOP paradigm.

13. Define Inheritance. Explain how to declare and initialise the Interface in java with a programming example [2016]

Ans.

An interface is basically a kind of class. Like classes, interface contain methods and variables but with a major difference. The difference is that interfaces define only abstract methods and final fields. This means that interfaces do not specify any code to implement these methods and data fields contain only constants. Therefore, it is the responsibility of the class that implements an interface to define the code for implementation of these methods.

- An interface a programming construct which contains only public abstract methods and public static final variables.
- Abstract method is the method which does not have body is called as an abstract method but that method only contains signature.
- An interface can extend multiple interfaces.
- An interface is fully an unimplemented structure in which none of the methods will have body.

➢ The general form of an interface is,

```
interface interface_name
{
    variables declaration;
    methods declaration;
}
```

➢ Extending of an interface can be done as follows,

```
interface i1
{
    void method 1();
}
```

```
interface i2 extends i1
{
    void method 2();
}
```

- The general syntax for implementing an interface is,

```
class classname implements interfacename
{
    //body of the class
        }
```

- Syntax for implementing more than one interface is,

```
class classname implements interfacename1,interfacename2, ...
{
    //body of the class
}
```

Example:

```
interface Area
{
    final static float pi=3.14F;
    float compute (float x, float y)
}
class Rectangle implements Area
{
    public float compute (float x, float y)
    {
        return(x*y);
    }
}

class Circle implements Area
{
    public float compute (float x, float y)
    {
        return(pi*x*x);
    }
}

class InterfaceTest
{
    public static void main(String args[])
    {
        Rectangle rect=new Rectangle();
        Circle cir=new Circle();
        Area area;
        System.out.println("Area of Rectangle="+area.compute(10,20));
```

```
            area=cir;
            System.out.println("Area of Circle="+area.compute(10,0));
        }
    }
```

14. Explain Different ways of creating threads in JAVA [2016]

Ans.

A thread is similar to a program that has a single flow of control. It has a beginning, a body, and an end, and executes commands sequentially. Every program will have at least one thread. A thread is a part of a program that is running. A thread is a single sequential flow of control within a process. A thread is also referred to as a light weight process. Executing two or more threads at the same time is called as multithreading.

There are two types of threads:

i. Demon Thread (created by JVM)
ii. User Thread (created by programmer)

main() thread is automatically created by JVM. Thread based is multi tasking is single program performing more than one task at the same time.

Creating Threads:

Creating threads in Java is simple. Threads are implemented in the form of objects that contain a method called run(). The run() method is the heart and soul of any thread. It makes up the entire body of a thread and is the only method in which the thread's behaviour can be implemented. A typical run() would appear as follows:

```
public void run()
{
    //statements for implementing thread
}
```

The run() method should be invoked by an object of the concerned thread. This can be achieved by creating the thread and initiating it with the help of another thread method called start().

A new thread can be created in two ways.

1) By creating a thread class: Define a class that extends Thread class and override its run() method with the code required by the thread.
2) By converting a class to a thread: Define a class that implements Runnable interface. The Runnable interface has only method, run(), that is to be defined in the method with the code to be executed by the thread.

The approach to be used depends on what the class we are creating requires. If it requires to extend another class, then we have no choice but to implement the Runnable interface, since Java classes cannot have two superclasses.

Example:

```
class A extends Thread
{
```

```
        public void run()
        {
                for(int i=1;i<=5;i++)
                {
                        System.out.println("From Thread A:i="+i);
                }
                System.out.println("Exit from A");
        }
}
class B extends Thread
{
        public void run()
        {
                for(int j=1;j<=5;j++)
                {
                        System.out.prinln("From Thread B:j="+j);
                }
                System.out.println("Exit from B");
        }
}
class C extends Thread
{
        public void run()
        {
                for(int k=1;k<=5;k++)
                {
                        System.out.println("From Thread C:k="+k);
                }
                System.out.println("Exit from C");
        }
}
class ThreadTest
{
        public static void main(String args[])
        {
                new A().start();
                new B().start();
                new C().start();
        }
}
```

15. Write a note on:

a. **Producer consumer problems**

b. **Read write problem** **[2016]**

Ans.

a) Producer Consumer Problem:

In computing, the producer consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two

processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

- The producer's job is to generate data, put it into the buffer, and start again.
- At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

Problem:

To make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

Solution:

The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

An inadequate solution could result in a deadlock where both processes are waiting to be awakened.

- In PC Class (A class that has both produce and consume methods), a linked list of jobs and a capacity of the list is added to check that producer does not produce if the list is full.
- In Producer class, the value is initialized as 0.
  - Also, we have an infinite outer loop to insert values in the list. Inside this loop, we have a synchronized block so that only a producer or a consumer thread runs at a time.
  - An inner loop is there before adding the jobs to list that checks if the job list is full, the producer thread gives up the intrinsic lock on PC and goes on the waiting state.
  - If the list is empty, the control passes to below the loop and it adds a value in the list.
- In the Consumer class, we again have an infinite loop to extract a value from the list.
  - Inside, we also have an inner loop which checks if the list is empty.
  - If it is empty then we make the consumer thread give up the lock on PC and passes the control to producer thread for producing more jobs.
  - If the list is not empty, we go round the loop and removes an item from the list.
  - In both the methods, we use notify at the end of all statements. The reason is simple, once you have something in list, you can have the consumer thread consume it, or if you have consumed something, you can have the producer produce something.
  - sleep, at the end of both methods just make the output of program run in step wise manner and not display everything all at once so that you can see what actually is happening in the program.

b) Read Write Problem:

Synchronization for Readers/Writers Problem.

- An object is shared among may threads, each belonging to one of two classes:
  - Readers: read data, never modify it

- Writers: read data and modify it
- Using a single lock on the data object is overly restrictive
  - Want many readers reading the object at once
  - Allow only one writer at any point
- Correctness criteria:
  - Each read or write of the shared data must happen within a critical section.
  - Guarantee mutual exclusion for writers.
  - Allow multiple readers to execute in the critical section at once.

- Implementation:
  - The first reader blocks if there is a writer, any other readers who try to enter block on mutex.
  - The last reader to exit signals a waiting writer.
  - When a writer exits, if there is both a reader and writer waiting, which goes next depends on the scheduler.
  - If a writer exits and a reader goes next, then all readers that are waiting will fall through (at least one is waiting on wrt and zero or more can be waiting on mutex).
- Alternative desirable semantics:
  - Let a writer enter its critical section as soon as possible.

16. Briefly explain different event listener interface methods [2016]

Ans.

The delegation event model has two parts: sources and listeners. Listeners are created by implementing one or more of the interfaces defined by the ' java.awt.event ' package. When an event occurs, the event source invokes the appropriate method defined by the listener and provides an event object as its arguments.

The ActionListener Interface
This interface defines the actionPerformed() method that is invoked when an action event occurs. Its general form is,
void actionPerformed(ActionEvent ae)

Interface	Description
ActionListener	Defines one method to receive action events.
AdjustmentListener	Defines one method to receive adjustment events.
ComponentListener	Defines four methods to recognise when a component is hidden, moved, resized, or shown.
ContainerListener	Defines two methods to recognise when a component is added to or removed from a container.
FocusListener	Defines two methods to recognise when a component gains or losses keyboard focus.

ItemListener	Defines one method to recognise when the state of an item changes.
KeyListener	Defines three methods to recognise when a key is pressed, released, or typed.
MouseListener	Defines five methods to recognise when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
MouseMotionListener	Defines two methods to recognise when the mouse is dragged or moved.
MouseWheelListener	Defines one method to recognise when the mouse wheel is moved.
TextListener	Defines one method to recognize when a text value changes.
WindowFocusListener	Defines two methods to recognise when a window gains or losses input focus.
WindowListener	Defines seven methods to recognise when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.