

UNIT1

2 Marks

1) What is Java?

- Java is an object-oriented computer language.
- It is a high-level programming language developed by James Gosling in Sun Microsystems in 1995.
- Java is a fast, secure and reliable language used for many games, devices and applications.

2) Outline the major Java features.

- **Object-Oriented** – java is based on object-oriented programming where the class and methods describe about the state and behavior of object.
- **Portable** – Java program gets converted into Java Byte Codes that can be executed on any platform without any dependency.
- **Platform independent** – java works on “write once and run anywhere” as it supports multiple platforms like Windows, Linux, Mac, Sun Solaris, etc.
- **Robust** – Java has a strong memory management as there is no pointer allocation. It has automatic garbage collection that prohibits memory leaks.
- **Interpreted** – java compiler converts the codes into Java Byte Codes which are then interpreted and executed by Java Interpreter.

3) What is difference between Java and c?

C Programming	Java Programming
It requires that the functions with no arguments, with the void keyword	It requires that the functions with no arguments must be declared with empty parenthesis, not with the void keyword
C has no operators such as instanceof and >>>.	Java adds new operators such as instanceof and >>>.
C adds have a break and continue statements.	Java adds labeled break and continue statements.
C has no object-oriented programming features.	Java adds many features required for object-oriented programming.

4) What is difference between Java and C++?

C++ Programming	Java Programming
It support operator overloading.	It does not support operator overloading.
It support has template classes.	It does not have template classes as in C++.
It supports multiple inheritances of classes.	It does not support multiple inheritances of classes. This is accomplished using a new feature called "Interface".
It supports global variables.	It does not support global variables. Every variable and method is declared within classes and forms part of that class.
It supports pointers.	It does not use pointers.
It does not support destructor function with a finalize() function.	It has replaced the destructor function with a finalize() function.
There are header files in Java.	There are no header files in Java.

5) What is Web browser? Give examples.

- The browser application retrieves or fetches code, usually written in HTML (Hypertext Markup Language) and/or another language, from a web server, interprets this code, and renders (displays) it as a Web page for you to view. on the computer or another Internet-enabled device that supports a browser. **An example of Web Browsers:**
Hot Java, Netscape Navigator, Internet Explorer, Google Chrome

6) What is jdk?

- Java Development Kit:
Java Development Kit Comes with a collection of tools that are used for developing and Running java programs.
- Javac :- Compiler used for Compiling java programs.
- Java :- For Interpreting java programs.
- Appletviewer :- For Running Applets.
- Javadoc :-Creates HTML format documentation from java source code file.
- Javah:-Produces header files for use with native methods.
- Javap :- Java disassembler, which enable us to convert byte code files into a program description.
- jdb:- java debugger, which helps us to fins errors in our program.

7) Write Java program to calculate the square root of a number.

```

import java.lang.Math; class
SquareRoot
{
    public static void main(String args[])
    {
        double x = 5; //Declaration and initialization double
        y; //Simple declaration y=Math.sqrt(x);
        System.out.println("Y = "+y);

    }
}

```

8) What is Java Tokens? List Java Tokens.

Tokens are the Smallest unit of Program There are Five Types of Tokens. Reserve Word or Keywords, Identifier, Literals, Operators, Separators

9) What is Literals? What are the types of Literals?

Literals

Literals in java are sequence of characters (digits, letters and other characters) that represent constant values to be stored in variables. Java language specifies five major type of literals.

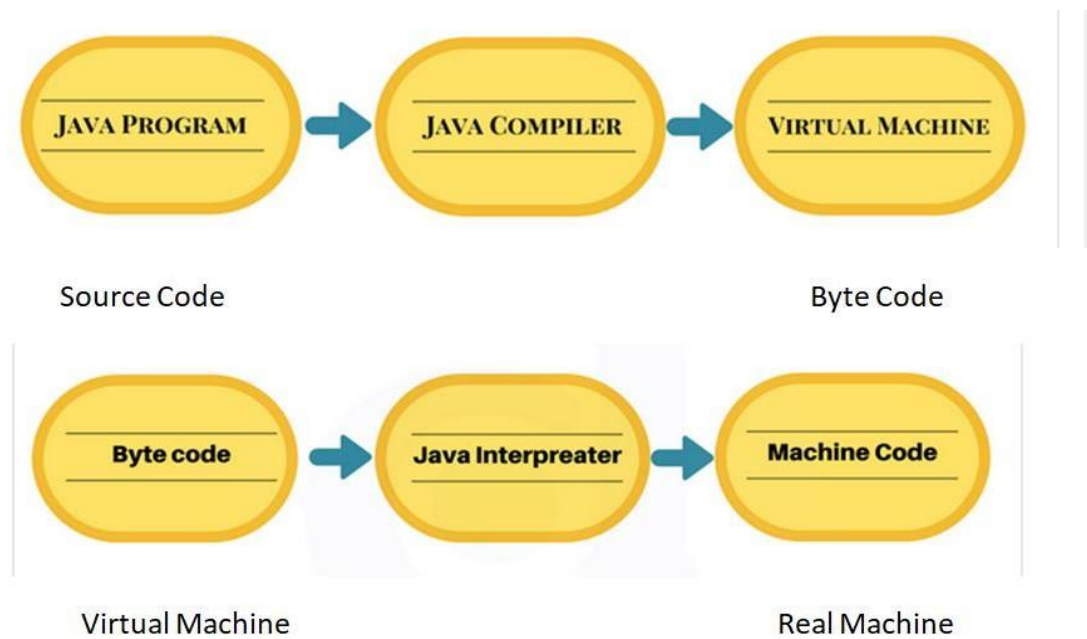
They are below :

Integer Literals Literals which stores integer value Floating
Literals Literals which stores float value Character Literals
Literals which stores character value String Literals Literals
which stores string value Boolean Literals Literals which stores
true or false

10) What is JVM?

JAVA VIRTUAL MACHINE

- Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM).



11) What is a Command Line argument?

- Command Line Argument is the parameters or arguments passed to the program when you run the program.
- The passed parameters is stored as a string array in the main method `public static void main(String args[])`
- Example

```
C:\jdk1.8.0_131\bin>java ComLineEx Simple Object-Oriented Robust Secure
```

12) **What is Constant? What are the types of constants in java?** Constants in java are fixed values those are not changed during the Execution of program java supports several types of Constants those are:

- a. Integer Constants
- b. Real Constants
- c. Single Character Constants
- d. String Constants
- e. Backslash Character Constants

13) What are variables in Java? List some rules for naming variables?

Variable is an identifier that denotes a storage location used to store a value.

Rules for naming variables:

- They must not begin with a digit.
- Uppercase and Lowercase are distinct. This means that the variable Total is not the same as total or TOTAL.
- It should not be keyword.
- White spaces is not allowed.
- Variables name can be of any length

14) What are the types of variables in Java?

Instance Variables

The Variables those are declared in a class are known as instance variables When object of Class is created then instance Variables are Created So they always Related With Class Object But Remember Only one memory location is created for one instance variable.

Local Variables

The Variables those are declared in Method of class are known as Local Variables

Class Variables

The Variables those are declared inside a class are called as Class variables or also called as data members of the class.

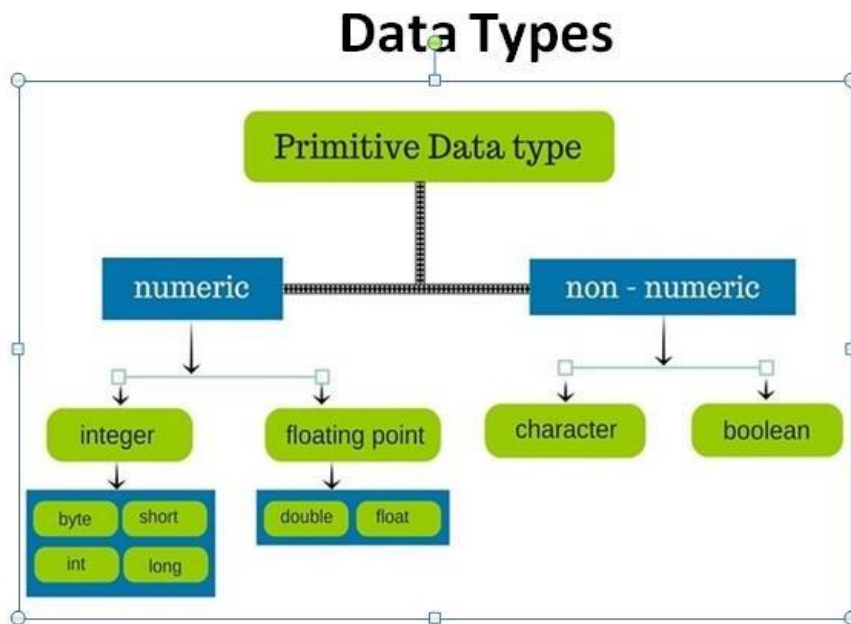
15) What is datatype?what are the different datatypes used in Java?

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

Primitive data types:

These are the predefined data types. In java there are 8 primitive data types which are as follows:

Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays.



16) What is Typecasting? Give Example.

Type casting is used to convert an object or variable of one type into another.

Syntax

```
dataType variableName = (dataType) variableToConvert;
```

Example :

```
int x = 10;  
byte y = (byte)x;
```

17) What is implicit and explicit typecasting?

Implicit Typecasting

- The conversion of a data type which is carried out automatically by the compiler without programmer intervention is called the implicit type conversion.
- Converts the lower data type to higher data type.

Explicit Typecasting

- The conversion of a data type which is carried out by the programmer is called the explicit type conversion.
- Converts the higher data type to lower data type.

18) What is Operators? List Operators in Java.

Operators

Operators are the Special Symbols those have specific functions associated with them for performing the operations it needs some Operands. Operands are those on which operations are performed by the Operators Like 2 +3

In this + is the Operator and 2 and 3 Operands.

The types of Operators those are supported by Java Language.

- | | |
|--|--------------------------|
| 1. Assignment Operators | 2. Arithmetic Operators |
| 3. Relational or Conditional Operators | 4. Logical Operators |
| 5. Increment Decrement Operators | 6. Conditional Operators |
| 7. Special Operators | 8. Dot Operator |

19) What is Conditional operator in Java?

This operator is used When we wants to specify the value of variable by first checking a condition This Operator Works Similar to if else :- In this First it checks the condition if it is true then it will gives the result and if a condition will false then it also Gives Some Values to the Variables

This Operator Contains ? :

For Ex:

Int A,B

A=10

B=20

C=(A>B) ? 13: 20

20) What is the difference between typecasting and type conversion? Type conversion is done “automatically” by compiler from lower data type to higher data type whereas, type casting is to be “explicitly done” by the programmer.

21) List different types of decision making statements in Java.

- if statement
- if-else statement
- nested if statement
- if-else-if ladder
- switch

22) Give syntax of nested if statement.

```
If(test condition1 )
{
    if(test condition2)
    {
        statement-1;
    }
    else
    {
        statement-2;
    }
else
    {
        statement-3;
    }
}
```

23) **Give syntax of switch statement.**

```

Statementr-x;
switch (expression)
{
    case value1:
        block-1
        break;
    case value2:
        block-2
        break;
    .....
    ..... default:
        default-block break;
}
Statement-x;

```

24) What is Looping? What is the basic operation performed in Loop?

The Process of Repeatedly performing tasks is known as looping In the loop generally there is three basic operations are performed

- 1) Initialization
- 2) Condition check
- 3) Increment

25) What is Entry Controlled and Exit Controlled loop? Give example. First Checks Condition for Execution then it is called as Entry Controlled Loop and if a Loop Checks Condition after the Execution of Statement then they are called as Exit Controlled Loops.

Entry Controlled: - while loop, for loop
Exit Controlled:- do-while loop

26) What is the difference between print() and println()?

The print() method prints output on the line until a new character is encountered. For example, the statements

Ex
System.out.print("Let us"); System.out.print("Learn
Java");

o/p- Let us Learn Java

The println() method take the information provided and displays it on a line followed by a line feed (carriage return). Therefore, the statements System.out.println("Let us"); System.out.println("Learn Java"); will

display the following output:

Let us Learn Java

5 Marks

1) Write short note on Java History.

Year	Development
1990	Sun decided to developed special software that could be used for electronic devices. A project called Green Project created and head by James Gosling.
1991	Explored possibility of using C++, with some updates announced a new language named "Oak"
1992	The team demonstrated the application of their new language to control a list of home appliances using a hand held device.
1993	The World Wide Web appeared on the Internet and transformed the text-based interface to a graphical rich environment. The team developed Web applets (time programs) that could run on all types of computers connected to the Internet.
1994	The team developed a new Web browsed called "Hot Java" to locate and run Applets. HotJava gained instance success.
1995	Oak was renamed to Java, as it did not survive "legal" registration. Many companies such as Netscape and Microsoft announced their support for Java
1996	Java established itself it self as both 1. "the language for Internet programming" 2. a general purpose OO language.
1997	Sun releases Java Development Kit(JDK1.1)

2) Write and explain any five features in java.

a) **Compiled and Interpreter**

Computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system. In first stage Java compiler (javac) translates source code into bytecode instructions. Bytecode are not machine instruction therefore, in the second stage java interpreter converts bytecode to machine code.

So java is compiled and interpreted language.

b) **Platform Independent**

Java is platform independent(that is, architecture-neutral), meaning that you can write a program once and run it on any computer.

Java ensures portability in two ways. First ,Java compile generates bytecode that can be implemented on any machine. Secondly ,the size of the primitive data types are machine independent

c) **Object-Oriented**

Java is purely OOP Language that is all the Code of the java Language is Written into the classes and Objects So For This feature java is Most Popular Language because it also Supports Code Reusability, Maintainability etc

d) **Robust**

The Code of java is Robust means first checks the reliability of the code before execution when we trying to Convert the Higher data type into the Lower then it Checks the Demotion of the code then It will warns a user to not to do this So it is called as Robust

e) **Simple Small and Familiar**

Java is a simple language because it contains many features of other Languages like c and C++ and Java removes complexity because it doesn't use pointers, storage classes and go to statements and java doesn't support Multiple Inheritance.

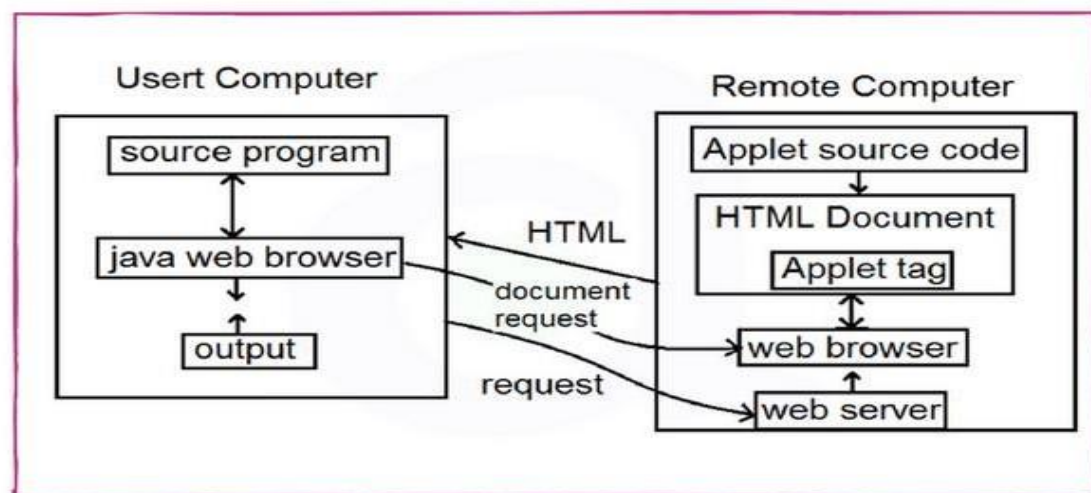
3) What is difference between C++ and Java?

Difference between JAVA and C++

C++ Programming	Java Programming
It support operator overloading.	It does not support operator overloading.
It support has template classes.	It does not have template classes as in C++.
It supports multiple inheritances of classes.	It does not support multiple inheritances of classes. This is accomplished using a new feature called "Interface".
It supports global variables.	It does not support global variables. Every variable and method is declared within classes and forms part of that class.
It supports pointers.	It does not use pointers.
It does not support destructor function with a finalize() function.	It has replaced the destructor function with a finalize() function.
There are header files in Java.	There are no header files in Java.

Activate Windows
Go to Settings to activate Windows.

4) Explain java's interaction with web.



Java and World Wide Web (www)

1. Java communicates with a web page through a special tag called <applet>.
2. Java user sends a request for an HTML document to the remote computers net browser.
3. The web-browser is a program that accepts a request, processes the request and sends the required documents.
4. The HTML document is returned to that user browser.
5. The document contains the applet tag which identifies the applet.
6. The corresponding applet is transferred to the user computer.
7. The Java enabled browser on the user's computer interprets the byte code and provide output.

5) Explain simple java program line by line.

//A Simple Example of java Program class

Sampleone

```
{
    public static void main(String arg[])
    {
        System.out.print("A Simple Example of java Program");
    }
}
```

- Comment

The first line is a single line comment which will be ignored during compilation. Comments make you understand what your program does. Use /* */ for multiple line comments.

- Class Declaration

In Java, everything must be declared in a class, which is declared by the keyword class. public is a keyword which makes this class available and accessible to anyone.

- Opening Brace

In Java, every class definition begins with an opening brace "{" and ends with a closing brace "}". Pair of "{}" braces define the scope of any method or class.

- The Main Line

The next line of code is shown here:

```
Public static void main(String arg[])
```

Public Here Public is a Keyword which declares main method is public or unprotected so that it will be Accessible to all classes.

Static Static is used for Describing that this is the method which belongs to this class not to other means this is the main method of Class only **Main** Method Must be Declared as Static so that Interpreter Will Execute this Method First ,before Creating the Objects of the Class.

Void Void means The Main Function will never Return a Value or it will Simply Prints the text on the Screen

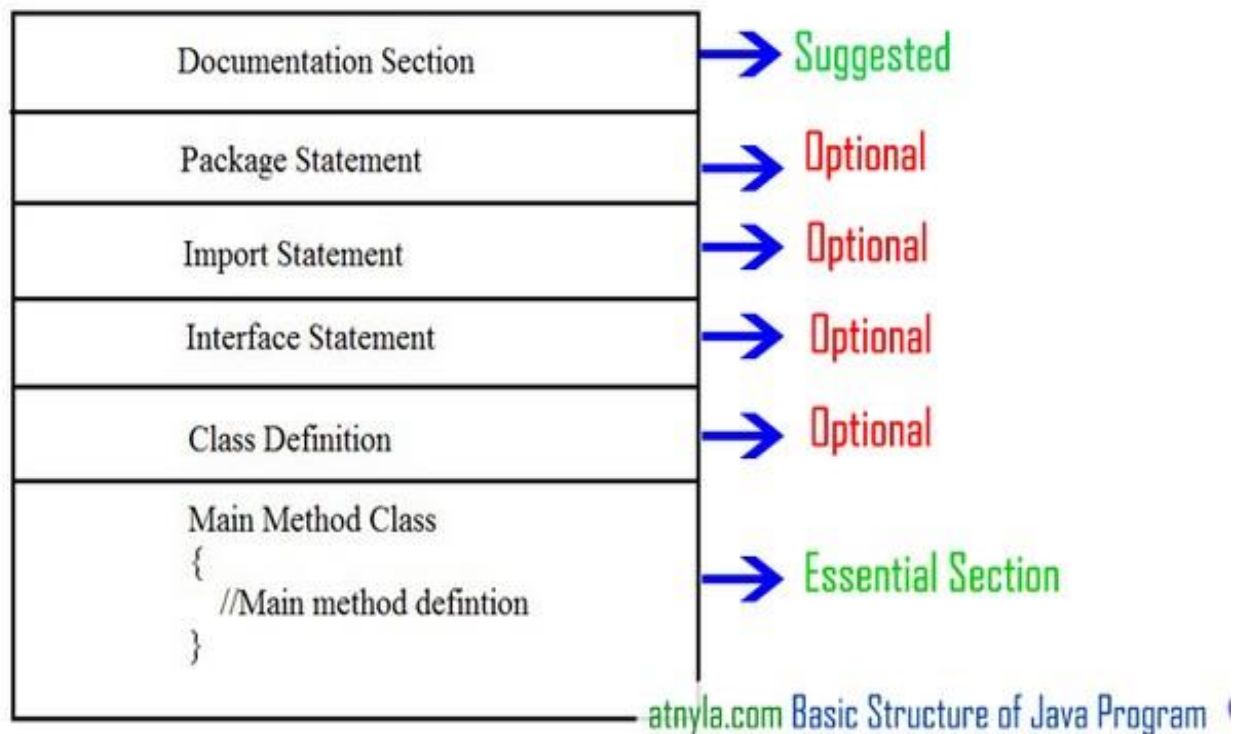
- The Output Line

The next line of code is shown here:

System.out.print("A Simple Example of java Program");

This Line is Similar to printf in C and Cout in C++ For displaying the Results on the Screen at the Time of Execution in this System is name of Package and Out i.e Output is name of class and print is the name of Method that belongs to the output class For Displaying the Results on the Screen.

6) Explain java program structure.



Documentation Section

It includes the comments that improve the readability of the program.

A comment is a non-executable statement that helps to read and understand a program especially when your programs get more complex.

Single line (or end-of line) comment:

```
// Calculate sum of two numbers
```

Multiline Comment:

```
/*calculate sum of two numbers and it is a          multiline  
comment */
```

Package Statement

- A package is a collection of classes, interfaces and sub-packages. A sub package contains collection of classes, interfaces and sub-sub packages etc.
- `java.lang.*`; package is imported by default and this package is known as default package. It must appear as the first statement in the source code file before any class or interface declaration. This statement is optional.

Import statements

- Java contains many predefined classes that are stored into packages. In order to refer these standard predefined classes in your program, you need to use fully qualified name (i.e. `Packagename.className`).
- `import java.util.Date`

Interface Section

- In the interface section, we specify the interfaces. An interface is similar to a class but contains only constants and method declarations.

Class Definition:

- Java program may contain multiple class definition. Classes are primary feature of Java program. The classes are used to map real world problems.

Main Method Class Section:

The main method can create objects, evaluate expressions, and invoke other methods and much more. On reaching the end of main, the program terminates and control passes back to the operating system.

7) **List and explain Java Tokens.**

Tokens are the smallest unit of Program There are Five Types of Tokens.

Reserve Word or Keywords

Identifier

Literals Operators

Separators

Keywords

Keywords are a special part of a language definition. Keywords are predefined reserved words used in programming that have special meanings to the compiler.

Ex:int , if ,Boolean,void etc.

Identifiers

Identifiers are the names of variables, methods, classes, packages and interfaces

Examples

Employee ,Area ,Animal, MotorCycle, Furniture,Length etc.

Literals

Literals in java are sequence of characters (digits, letters and other characters) that represent constant values to be stored in variables. Java language specifies five major type of literals.

They are below:

Integer Literals, Floating Literals, Character Literals, String Literals, Boolean Literals.

Separators

Separators are symbols used to indicate where groups of code are divided and arranged. They basically define the shape and function of our code.

Ex:{ } , () , [] , ; , .

8) **List and explain Java statements.**

The Empty Statement

The empty statement doesn't do anything, but the syntax is ccasionally useful.

For example

```
for(int i = 0; i < 10; i++) // Increment array elements
```

```
{
```

```
/* empty */; // Loop body is empty statement
```

```
}
```

Labeled Statements

Any statement may begin with a label. Such labels must not be keywords. In Java are used as the argument of jump statements Example:

```
rowLoop: for(int r = 0; r < rows.length; r++)
    { // A labeled loop
        break rowLoop; // Use a label
    }
```

Expression Statements

Most statements are Expression statements.

Java has seven types of Expression statements. Assignment, Pre-Increment , Post-Increment, Pre –Decrement, Post-Decrement, Method call and Allocation Expression.

For Example

```
a = 1; // Assignment i++; //
Post-increment
--c; // Pre-decrement
```

Selection Statements

These select one of several control flows. There are three types of selection statements in Java : if,if-else and switch

For Example

```
if (username == null) // If username is null
{
    username = "John Doe"; // define it.
}
```

Iteration Statements

These specify how and when looping will take place. There are three types of iteration statements : while,do and for.

For Example

```
int count = 0; while
(count < 10)
{
    System.out.println(count);
    count++;
}
```

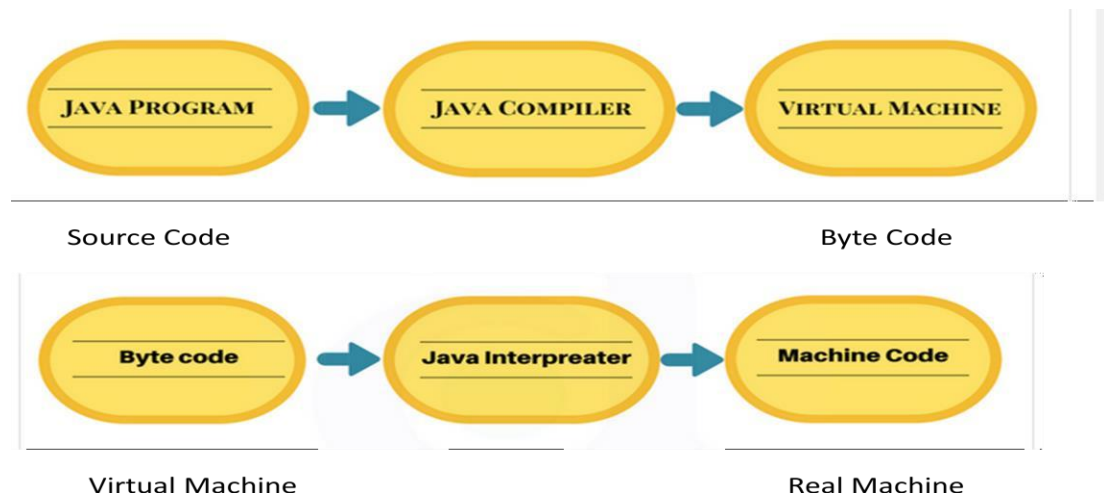
Jump Statements

These statements transfer control to another part of the program. Java supports four jump statements: break, continue, return and throw.

9) Write short note on JVM.

The Java language is a high-level language.

- All language compilers translate source code into machine code.
- Java compiler does the same thing. You save a Java program in a .java file and compile it into a .class file. The .class file (also called **bytecode**). Output of a Java compiler is not executable code. Rather, it is bytecode.
- Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM).
- The bytecode is similar to machine instructions but is architecture neutral and can run on any platform that has a Java Virtual Machine (JVM).
- The JVM executes our code along with the code in the library.



10) What is a command line arguments? Write a program to demonstrate command line arguments.

- Command Line Argument is the parameters or arguments passed to the program when you run the program.
- The passed parameters is stored as a string array in the main method.

```
class ComLineEx
{
    public static void main (String args[])
    {
        for(int i=0;i<args.length;i++)
        {
```

```

        System.out.println("Java is : "+args[i]);
    }
}
}

```

Output:

C:\jdk1.8.0_131\bin>javac ComLineEx.java

C:\jdk1.8.0_131\bin>java ComLineEx Simple Object-Oriented Robust Secure

Java is : Simple

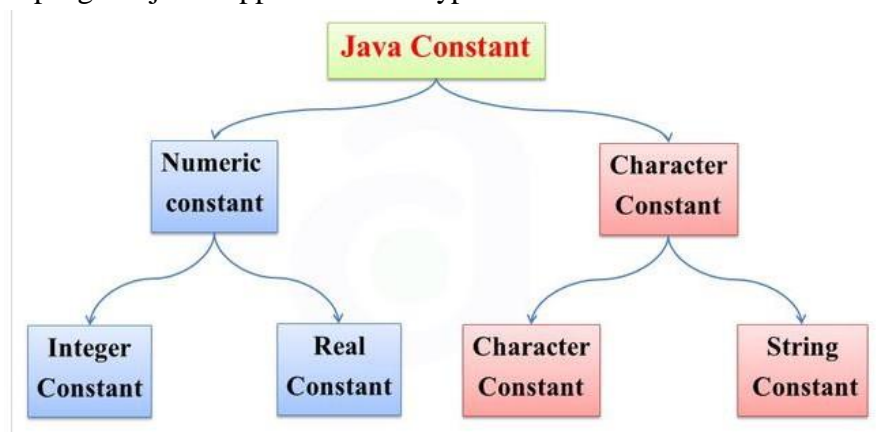
Java is : Object-Oriented

Java is : Robust

Java is : Secure

11) What are constants? Explain types of constants in java.

- Constants in java are fixed values those are not changed during the Execution of program java supports several types of Constants those are:



Integer Constants

Integer Constants refers to a Sequence of digits which Includes only negative or positive Values and many other things those are as follows Decimal Integer Constants

- Decimal constant consists of a set of digits 0 through 9, preceded by an optional – or + sign.
- 1, 3, 7, 8, 65, 543676664

Octal Integer Constants

An octal integer constant consists of any combination of digits from the set 0 through 7, with an leading 0.

- 038, 320, 0456, 0552, 0432

Hexadecimal Integer Constants

- An octal integer constant consists of any combination of digits from the set 0 through F, with an leading 0x or 0X.
- 0x4, 0X456, 0x552, 0x32, 0x43

Real Constants

- Integer numbers are unable to represent distance, height, temperature, price, and so on. These information are containing fractional parts or real parts like 56.890. Such numbers are called Real or Floating Point Constants
- For Example
- 0.0083, -2.13, 234.890

Single Character Constants

- A Character is Single Alphabet a single digit or a Single Symbol that is enclosed within Single inverted commas.
- Ex- 'a', 'A', '1', '4343', '#', '-', '<', 'X'

String Constants

- String is a Sequence of Characters Enclosed between double Quotes These Characters may be digits, Alphabets Like "Hello", "1234" etc.
- Ex – "a", "Ali", "123", "How are You"

12) Explain type casting in Java.

- Type casting is used to convert an object or variable of one type into another.

Syntax

`dataType variableName = (dataType) variableToConvert;`

Example :


```
int x = 10;
byte y = (byte)x;
```

In Java, type casting is classified into two types,

- Widening Casting(Implicit)

Automatic Type casting take place when, the two types are compatible the target type is larger than the source type

byte → short → int → long → float → double



widening

Example int

```
i = 100; long
```

```
l = i;
```

- Narrowing Casting(Explicitly done)

When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.



```
double d = 100.04;
long l = (long)d;
```

13) Explain operators in Java.

Assignment Operator

Assignment Operator (=) is used for Initializing a value to variable or we can say that an assignment operator is used for Assigning a value to the variable.

for Ex: a=10

Arithmetic Operators

The Arithmetic Operators are used for Performing the Mathematical Operations on operands Like + , - , / , * , %

Relational Operators

The Relational Operators are used for checking the Conditions or these operators are used for controlling the flow of the Execution to First Check the Condition it contains.

Operator	Result
<	Less Than
>	Greater Then
>=	Greater then Equals to
<=	Less Than Equals To
!=	not equal to

Logical Operators

The boolean logical operators shown here operate only on boolean operands. All of the binary logical operators combine two boolean values to form a resultant boolean value.

Operator	Result
&&	Logical AND
	Logical OR
!	Logical Not

Increment Decrement Operators

Increment/Decrement Operators is Counter Operator. These are two as: ++ (increment operator) and -- (decrement operator).

Increment operator are used for incrementing the value one by one. Similarly decrement operator are used for decrementing the value one by one. These are further sub-divided into two categories:

- a) Prefix Increment / Decrement Operator
- b) Postfix Increment / Decrement Operator

Conditional Operator

This operator is used When we wants to specify the value of variable by first checking a condition This Operator Works Similar to

if else :- This Operator Contains ? :

For Ex:

Int A,B

A=10

B=20

C=(A>B) ? 13 : 20

Special Operators

Java Also Provides Some Special Operators Like Instance of For checking Whether an Object is Instance of a Class or Whether an object is related to Class this will gives result true or either False

Dot Operator is used for Accessing the data and Member Function of class and This is used by Object of Class An Object Communicates With the Member Functions of Class With the Help of Dot Operator.

14) What is type conversion in java? Demonstrate one program on type conversion.

- The Type Conversion is that which automatically converts the one data type into another but remember we can't store a large data type into the small

For ex we can't store a float into int because a float is greater than int

- Type Conversion is also called as Promotion of data because we are converting one lower data type into higher data type So this is performed automatically by java compilers.

Example:

```
class Test
```

```
{  
    public static void main(String[] args)  
    {  
        int i = 100;  
  
        //automatic type conversion long  
        l = i;  
        //automatic type conversion float f = l;  
        System.out.println("Int value "+i);  
        System.out.println("Long value "+l);  
        System.out.println("Float value "+f);  
    }  
}
```

Output:

```
Int value 100  
Long value 100  
Float value 100.0
```

15) Write syntax and program on if statement.

Syntax of if statement

```
if (boolean-expression)  
{  
    statement(s);  
}
```

Program on if statement class

```
IfStatementExample
```

```
{  
    public static void main(String[] args)  
    {  
        float radius, area, PI;  
        PI=3.14f;
```



```

radius = 2.1f ; if
(radius >= 0)
{
    area = radius * radius * PI;
    System.out.println("The area for the circle of radius "
+radius
    +" is "+area);
}
}

```

16) Write syntax and program on nested-if.

- The syntax for a nested if...else is as follows –

```

If(test condition1 )
{
    if(test condition2)
    {
statement-1;

    }
else
    {
statement-2;

    }
else
    {
statement-3;

    }
}

```

Statement-r-x;

Program

```
class ifElseNesting
{
    public static void main(String arg[])
    {
        int a = 325,b=712,c=478;
        System.out.println("Largest value is : ");
        if ( a > b )
        {
            if (a > c)
            {
                System.out.println(a);
            }
            else
            {
                System.out.println(c);
            }
        }
        else
        {
            if( c > b)
            {
                System.out.println(c);
            }
            else
            {
                System.out.println(b);
            }
        }
    }
}
```

17) Write syntax and program on if-else-ladder.

Syntax

```
if(condition1)

    statement1; else if(condition2)
        statement2;
else if(condition3)
    statement3;

.....
else
    default-statement;
```

Example

class ElseifLadder

```
{
    public static void main(String args[])
    {
        int rollnumber [] = { 111,222,333,444}; int
        marks – { 81,75,43,58};
        for(int i=0;i<rollNumber.length;i++)
        {
            if (marks[i] > 79)
                System.out.println(rollnumber[i] + ““Honors”); else if
            (marks[i] > 59)
                System.out.println(rollnumber[i] + ““I Division”); else if
            (marks[i] > 49)
                System.out.println(rollnumber[i] + ““II
Division”);
            else
                System.out.println(rollnumber[i] + ““FAIL”);
        }
    }
}
```

18) Explain switch statement with syntax and example in java.

- switch case have multiple choice for selection of the statements or we can switch case is a multiway branch statement.
- The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement.
- It adds an easy way to dispatch execution to different parts of your code based on the value of an expression.

Syntax

```
switch (expression)
{
    case value1:
        block-1
        break;
    case value2:
        block-2
        break;
    .....
    .....
    default:
        default-block
        break;
}
```

Statement-x;

Example

switch(ch)

```
{
    case 'A':
    {
        System.out.print("\n\tYour Choice is Red\n\n");
        break;
    }
    case 'B':
```

```

        {
            System.out.print("\n\tYour Choice is Blue\n\n");
            break;
        }
        case 'C':
        {
            System.out.print("\n\tYour Choice is Yellow\n\n");
            break;
        }
        default:
        {
            System.out.print("\n\tYour Choide is Wrong...\n\n");
            break;
        }
    }
}

```

19) Explain while loop with syntax and example.

- while Loop is Known as Entry Controlled Loop because in The while loop first we initialize the value of variable or Starting point of Execution and then we check the condition and if the condition is true then it will execute the statements and then after it increments or decrements the value of a variable.
- Syntax Initialization;

```

while(test condition)
{
    Body of the loop
}

```
- Example

```

int i=1;
while(i <=10)
{

```

```
        System.out.print("\n\tI = "+i); i++;  
    }
```

20) Explain do-while loop with syntax and example.

- This is Called as Exit Controlled Loop

In do while loop first it executes the statements and then it increments the value of a variable and then last it checks the condition So in this either the condition is true or not it Execute the statement at least one time.

- Syntax

Initialization:

```
    {  
        Body of the loop  
    }  
while(test condition);
```

- Example

```
int i=1;  
  
    do  
    {  
        System.out.print("\n\tI = "+i); i++;  
    }  
  
}
```

21) Explain for loop with syntax and example.

- for Loop is Known as Entry Controlled Loop
- In This loop all the basic operations like initialization, condition checking and incrementing or decrementing all these are performed in only one line. this is similar to the while loop for performing its execution but only different in its syntax.

- Syntax

```
for(Initialization:test condition;increment)
{
    Body of the loop
}
```

- Example

```
int i;
for(i =1; i <=10; i++)
{
    System.out.print("\n\tI = "+i);
}
```

Java Programming

Unit-2

Classes, Arrays, Strings and Vectors

2 Marks Questions

1. What is class? How the class is defined?

Class is a user defined data type with a template that serves to define its properties.

Basic form of class definition is

```
Class classname [extends superclassname]
{
    [fields declaration]
    [methods declaration]
}
```

Everything inside the square bracket is optional

Class name and super class name are any valid java identifiers.

2. What are objects? How object is created in java?

An object in Java is essentially a block of memory that contains space to store all the instance variables.

Objects in java are created using the new operator.

Example:

```
Rectangle rect1;
```

```
rect1=new Rectangle();
```

Or

```
Rectangle rect1=new Rectangle();
```

3. What are constructors? Which are the different types of constructors?

Java supports a special type of method called a constructor that enables an object to initialize itself when it is created.

Types of Java constructors

There are two types of constructors:

Default constructor (no-arg constructor)

Parameterized constructor

4. What are the rules for creating constructors?

Rules for creating Java constructor

There are basically three rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type. i.e Constructors do not have a return type — not even void
- Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.

5. What is method overloading and mention its advantages.

If a class has multiple methods having same name but different in parameters, it is known as Method

Overloading.

Advantage of method overloading

- The main advantage of this is cleanliness of code.
- Method overloading increases the readability of the program.
- Flexibility
- Overloaded methods give programmers the flexibility to call a similar method for different types of data.

6. What are static members?

- The **static keyword** is used in java mainly for memory management.
- It is used with variables, methods, blocks and nested class.
- It is a keyword that are used for share the same variable or method of a given class. This is used for a constant variable or a method that is the same for every instance of a class.
- No object needs to be created to use static variable or call static methods, just put the class name before the static variable or method to use them.

7. What is inheritance? Which are the types of inheritance?

The mechanism of deriving a new class from an old class is called inheritance.

Types of Inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance

8. What is method overriding? Mention its advantages.

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

Advantage of method overriding

- The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class code.
- This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it

9. What happens when we declare variable, method and class as final?

- If you make any variable as final, you cannot change the value of a final variable (It will be constant).
- When a method is declared with final keyword, it is called a final method. A final method cannot be overridden.
- If we declare class as final it cannot be inherited by other classes. We cannot extend a final class.

10.Explain the purpose of finalize () method in Java.

Sometimes an object will need to perform some action when it is destroyed. For example closing an open connection or releasing any resources held. To handle such situations, Java provides a mechanism called finalizer. Method is simply **finalize()**

It is similar to destructor in C++

11.List access modifiers in Java.

Java has four access modifiers:

1. public
2. default (Friendly)
3. private
4. Protected

12.Define array. How to create and declare an array.

Array is a group of continuous or related data items that share a common name.

Syntax:

```
array_name[value];
```

- Arrays in Java may be declared in two forms:

Form1

```
type arrayname[];
```

Form2

```
type [] arrayname[];
```

Example:

```
int          number[ ];  
float[ ]     marks;
```

13.What is String and how to declare strings?

String represents a sequence of characters.

Declaring string:

```
String stringName;
```

```
StringName = new String ("string");
```

Example:

```
String firstName;
```

```
firstName = new String("ABC");
```

14. List different string methods.

Method Call	Task performed
S2=s1.toLowerCase();	Converts the string s1 to all lowercase
S2=s1.toUpperCase();	Converts the string s1 to all Uppercase
S2=s1.replace('x', 'y');	Replace all appearance of x with y
S2= s1.trim();	Remove white spaces at the beginning and end of the string s1
S1.equals(s2);	Returns true if s1 is equal to s2
S1.equalsIgnoreCase(s2);	Returns true if s1=s2 ignoring the case of characters
S1.length();	Gives length of s1
S1.charAt(n);	Gives nth character of s1
S1.compareTo(s2)	Returns negative if s1<s2, positive if s1>s2 and
S1.concat(s2)	Concatenates s1 and s2

15. What are vectors? How they are created?

Vector class can be used to create a generic dynamic array known as vector that can hold objects of any type and any number.

Vectors are created as array as follows:

```
Vector intVect = new Vector( ); // declaring without size
```

```
Vector list = new Vector (3); // declaring with size
```

16. Mention vector advantages and disadvantages.

Advantages of vector over arrays

- It is convenient to use vectors to store objects.
- A vector can be used to store a list of objects that may vary in size.
- We can add and delete objects from the list as and when required.

Disadvantages of vector over arrays

- We cannot directly store simple data types in a vector.
- We can only store objects. Therefore, we need to convert simple types to objects. This can be done using the wrapper classes

17.What is wrapper class?

- Since, vectors cannot handle primitive data types like int, float, long ,char, and double.
- Primitive data types may be converted into object types by using the wrapper classes contained in the java.lang packages.
- The wrapper classes have a number of unique methods for handling primitive data types and objects.

5 and 10 Marks Questions

1. How classes are defined in Java give the syntax and explain with proper example.

Class is a user defined data type with a template that serves to define its properties.

Basic form of class definition is

Class classname [extends superclassname]

```
{  
    [fields declaration]  
    [methods declaration]  
}
```

Everything inside the square bracket is optional

Class name and super class name are any valid java identifiers.

Example:

```
class Rectangle  
{  
    int length,width; //Declaration of variables  
    void getData(int x, int y) //Declaration of method  
    {  
        length=x;  
        width=y;  
    }  
    int rectArea() //Definition of another method  
    {  
        int area=length*width;  
        return(area);  
    }  
}  
class RectArea // class withmain method  
{  
    public static void main(String args[])  
    {  
        int area1,area2;  
        Rectangle rect1=new Rectangle();  
        Rectangle rect2=new Rectangle();  
        rect1.length=15; //Accessing variables  
        rect1.width=10;  
        area1=rect1.length*rect1.width;  
        rect2.getData(20,12); //Accessing methods  
        area2=rect2.rectArea();  
        System.out.println("Area1="+area1);  
        System.out.println("Area1="+area2);  
    }  
}
```

```
}
```

Output:

```
C:\jdk1.8.0_131\bin>javac RectArea.java
```

```
C:\jdk1.8.0_131\bin>java RectArea
```

```
Area1=150
```

```
Area1=240
```

2. Explain constructors with example.

Java supports a special type of method called a constructor that enables an object to initialize itself when it is created.

Rules for creating Java constructor

There are basically three rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type. i.e Constructors do not have a return type — not even void
- Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.

Types of Java constructors

There are two types of constructors:

Default constructor (no-argument constructor)

Parameterized constructor

- **Example**

```
class Rectangle
{
    int length,width;
    Rectangle()           //default constructor
    {
        length=0;
        width=0;
    }
    Rectangle(int x,int y) //parameterized constructor
    {
        length=x;
        width=y;
    }
    void rectarea()
    {
        int area=length*width;
        System.out.println("The area of rectangle is="+area);
    }
}
class cons
{
    public static void main(String args[])
    {
        Rectangle rect1=new Rectangle(); //calling default constructor
        Rectangle rect2=new Rectangle(15,10); //calling parameterized constructor
        rect1.rectarea();
        rect2.rectarea();
    }
}
```

```
}
```

Output:

The area of rectangle is=0

The area of rectangle is=150

3. What is method overloading explain with example?

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

Advantage of method overloading

- The main advantage of this is cleanliness of code.
- Method overloading increases the readability of the program.
- Flexibility
- Overloaded methods give programmers the flexibility to call a similar method for different types of data.

Different ways to overload the method

There are two ways to overload the method in java

- By changing number of arguments
- By changing the data type

Example:

```
class OverloadDemo
{
    void method(int a)
    {
        System.out.println("a: " + a);
    }
    void method(int a, int b)
    {
        System.out.println("a and b: " + a + " " + b);
    }
    double method(double a)
    {
        System.out.println("double a: " + a);
        return a*a;
    }
}
class OverloadExample
{
    public static void main(String args[])
    {
        OverloadDemo ob = new OverloadDemo();
        double result;    // call all versions of test()
        ob.method(10);
        ob.method(10, 20);
        result = ob.method(123.25);
        System.out.println("Result of method(123.25): " + result);
    }
}
```

Output:

a: 10

a and b: 10 20

double a: 123.25

Result of mehod(123.25): 15190.5625

4. Explain static members with example.

- The **static keyword** is used in java mainly for memory management. It is used with variables, methods, blocks and nested class.
- No object needs to be created to use static variable or call static methods, just put the class name before the static variable or method to use them.

In java language static keyword can be used for following

- variable (also known as a class variable)
- method (also known as a class method)

Static variable

- If you declare any variable as static, it is known static variable.
- The static variable gets memory only once in a class area at the time of class loading.

Ex: static int count;

static method in java

- It is a method which belongs to the class and no need to create the object(instance)
- A static method can call only other static methods and cannot call a non-static method from it.
- A static method can be accessed directly by the class name and doesn't need any object

Ex: static int max(int x,int y);

Example:

```

class operation
{
    static float mul(float x,float y)
    {
        return x*y;
    }
    static float divide(float x,float y)
    {
        return x/y;
    }
}
class mathapplication
{
    public static void main(String args[])
    {
        float a=operation.mul(4,5);
        float b=operation.divide(a,2);
        System.out.println("b = "+b);
    }
}

```

Output:

b = 10.0

5. What is nesting of method? Give example.

A method can be called by using only its name by another method of the same class. This is known as nesting of method.

```
class Nesting
{
    int m,n;
    Nesting(int x,int y)
    {
        m=x;
        n=y;
    }
    int largest()
    {
        if(m>=n)
        return(m);
        else
        return(n);
    }
    void display()
    {
        int large=largest();
        System.out.println("Largest value="+large);
    }
}
class NestingTest
{
    public static void main(String args[])
    {
        Nesting nest=new Nesting(50,40);
        nest.display();
    }
}
```

Output:

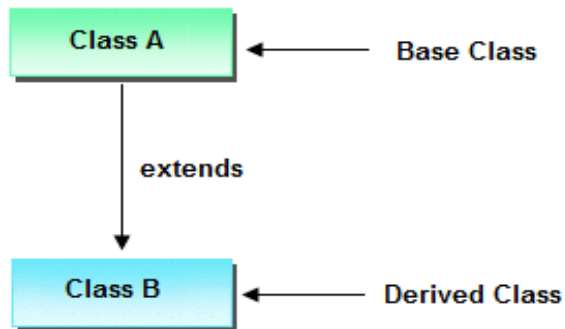
Largest value=50

6. Explain single inheritance in Java with example.

The mechanism of deriving a new class from an old class is called inheritance.

Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
    variable declaration;
    methods declaration ;
}
```



- **Single Inheritance:**
- When a class extends another one class only then we call it a single inheritance.

Class A

```
{
    public void methodA()
    {
        System.out.println("Base class method");
    }
}
```

Class B extends A

```
{
    public void methodB()
    {
        System.out.println("Child class method");
    }
}
```

Class single

```
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.methodA(); //calling super class method
        obj.methodB(); //calling local method
    }
}
```

Output:

Base class method

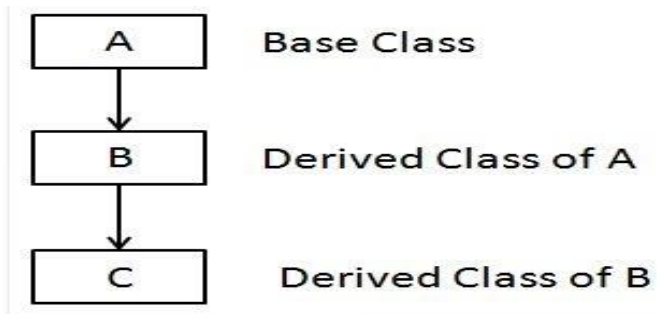
Child class method

7. Explain multilevel inheritance in Java with example.

The mechanism of deriving a new class from an old class is called inheritance.

- **Multilevel Inheritance**

Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class.



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");
    }
}
```

Class multi

```
{
    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
    }
}
```

Output:

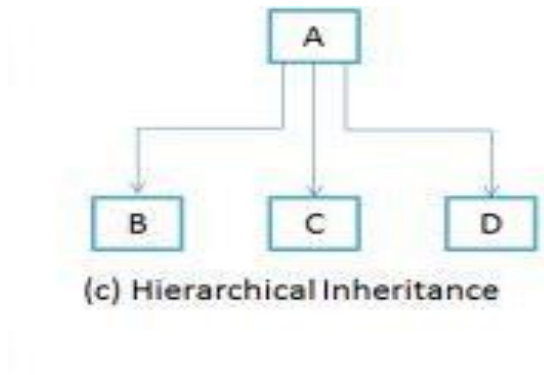
```
Class X method
class Y method
class Z method
```

8. Explain hierarchical inheritance in Java with example.

The mechanism of deriving a new class from an old class is called inheritance.

- **Hierarchical Inheritance**

It is type of inheritance in which one class is inherited by many **sub classes**.



```
class A
{
public void methodA()
{
System.out.println("method of Class A");
}
}
class B extends A
{
public void methodB()
{
System.out.println("method of Class B");
}
}
class C extends A
{
public void methodC()
{
System.out.println("method of Class C");
}
}
class JavaExample
{
public static void main(String args[])
{
B obj1 = new B();
C obj2 = new C(); //All classes can access the method of class A
obj1.methodA();
obj2.methodA();
}
}
```

Output:

method of Class A
method of Class A

9. Explain method overriding with example.

- If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.
- When method is called, the method defined in subclass is executed instead of method in superclass.

Advantage of method overriding

- The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class code.
- This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it

Example:

```
class A
{
    //Overridden method
    public void display()
    {
        System.out.println("Method in class A");
    }
}
class B extends A
{
    //Overriding method
    public void display()
    {
        System.out.println(" Method in class B");
    }
}
Class overridetest
{
    public static void main( String args[])
    {
        B obj = new B(); //This will call the child class version of display()
        obj.display();
    }
}
```

Output: Method in class B

10.Explain abstract class and method with example.

Abstract method is opposite to final method. If u define method as abstract that must always be redefined in a subclass thus making overriding compulsory.

- **Abstract class in Java**
- Abstract classes may or may not contain *abstract methods*, i.e. methods without body (public void get();)
- But, if a class contain at least one abstract method, then the class must be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To utilize an abstract class, you have to inherit it from another class, provide implementations for the abstract methods in it.

Ex: abstract class class_name { }

11.Explain string methods.

String class defines a number of methods that allows us to accomplish a variety of string manipulation tasks.

Method Call	Task performed
S2=s1.toLowerCase();	Converts the string s1 to all lowercase
S2=s1.toUpperCase();	Converts the string s1 to all Uppercase
S2=s1.replace('x', 'y');	Replace all appearance of x with y
S2= s1.trim();	Remove white spaces at the beginning and end of the string s1
S1.equals(s2);	Returns true if s1 is equal to s2
S1.equalsIgnoreCase(s2);	Returns true if s1=s2 ignoring the case of characters
S1.length();	Gives length of s1
S1.charAt(n);	Gives nth character of s1
S1.compareTo(s2)	Returns negative if s1<s2, positive if s1>s2 and
S1.concat(s2)	Concatenates s1 and s2

Example:

```
import java.lang.String;

public class java_9
{
    public static void main(String args[])
    {
        String s1,s2,s3;
        s1= "Hello";
        s2="World";
        s3=" ";
        s1=s1.toLowerCase();
        s2=s2.toUpperCase();
        s3=s1.concat(s2);
        int l= s3.length();
        char c=s3.charAt(5);

        System.out.println("To LowerCase : " + s1);
        System.out.println("To UpperCase : " + s2);
    }
}
```

```

        System.out.println("Concatneted string : " + s3);
        System.out.println("String Length : " + l);
        System.out.println("Char at 5 : " + s1);
    }
}

```

12.Explain String Buffer class with an example.

- String Buffer class: is a peer class of String. While String creates String of fixed length,
- String Buffer creates strings of flexible length that can be modified in terms of both length and content.
- We can insert characters and substrings in the middle of the string or append another string to end.

String Buffer Methods

Method	Description
setCharAt(n, 'x')	Modifies the nth character to x
append(s2)	Appends the string s2 to s1 at the end
insert(n,s2)	Inserts the string s2 at the position n of the string s1
setLength(n)	Sets the length of the string s1 to n .

Example:

```

StringBuffer sb= new StringBuffer("Hello");
sb.append("world");
System.out.println(sb);

```

Output:

Hello world

```

sb.insert(5, " ");
sb.insert(sb.length(),2019);
System.out.println(sb);

```

Output:

Hello world 2019

13.Explain different types of vector methods.

- Vectors Class: can be used to create a generic dynamic array known as vector that can hold objects of any type and any number.
- Vectors are created as array as follows:
Vector<Integer> vect = new Vector<>(); // declaring without size
Vector<Integer> list = new Vector<Integer>(3); // declaring with size
- Vector Methods:

Method Call	Task Performed
List.addElement(item)	Adds the item specified to the list at the end
List.elementAt(10)	Gives the name of the 10 th object
List.size()	Gives the number of objects present
List.removeElement(item)	Removes the specified item from the list
List.removeElementAt(n)	Removes the item stored in the nth position of the list
List.removeAllElements()	Remove all the elements in the list
List.copy Into(array)	Copies all items from list to array
List.insertElementAt(item, n)	Insert the item at nth position

14.What is wrapper class? Give an example.

- Since, vectors cannot handle primitive data types like int, float, long ,char, and double.
- Primitive data types may be converted into object types by using the wrapper classes contained in the java.lang packages.
- The wrapper classes have a number of unique methods for handling primitive data types and objects.

Simple Type	Wrapper Class
Boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long

Example1:

```
public class WrapperExample1
{
    public static void main(String args[])
    {
        //Converting int into Integer
        int a=20;
        Integer i=Integer.valueOf(a); //converting int into Integer
        Integer j=a; //autoboxing, now compiler will write Integer.valueOf(a) internally

        System.out.println(a+" " +i+ " " +j);
    }
}
```

Output

20 20 20

Example2:

```
public class WrapperExample2
{
public static void main(String args[])
{
//Converting Integer to int
Integer a=new Integer(3);
int i=a.intValue(); //converting Integer to int
int j=a;    //unboxing, now compiler will write a.intValue() internally
```

```
System.out.println(a+" "+i+" "+j);
```

```
}
```

```
}
```

Output

3 3 3

15.Program to demonstrate method overloading.

```
class OverloadDemo
{
    void test()
    {
        System.out.println("No parameters");
    }
    // Overload test for one integer parameter.
    void test(int a)
    {
        System.out.println("a: " + a);
    }
    // Overload test for two integer parameters.
    void test(int a, int b)
    {
        System.out.println("a and b: " + a + " " + b);
    }
    // overload test for a double parameter
    double test(double a)
    {
        System.out.println("double a: " + a);
        return a*a;
    }
}
class Overload
{
    public static void main(String args[])
    {
        OverloadDemo ob = new OverloadDemo();
```



```

        double result;
        // call all versions of test()
        ob.test();
        ob.test(10);
        ob.test(10, 20);
        result = ob.test(123.2);
        System.out.println("Result of ob.test(123.2): " + result);
    }
}

```

16. Program to demonstrate the method overriding

```

class Base
{
    void type()
    {
        System.out.println("This is base Class");
    }
}
class B1 extends Base
{
    void type()
    {
        System.out.println("This is Class B1");
    }
}
class B2 extends Base
{
    void type()
    {
        System.out.println("This is Class B2");
    }
}
class java13
{
    public static void main(String args[])
    {
        B1 b1 = new B1();
        B2 b2 = new B2();

        b1.type();
        b2.type();
    }
}

```

Java Programming

Unit-3

Interfaces, Packages and Multithreaded Programming

2 Marks Questions

1. What is interface? Give syntax to define interface.

Java programming language does not support multiple inheritances, using interfaces we can achieve concept of multiple inheritance.

Like classes, interfaces contain methods and variables but define only abstract methods and final fields.

Syntax:

```
interface interfacename
{
    variable declaration;
    methods declaration;
}
```

2. Differentiate between class and interface.

Class	Interface
The members of class can be constant or variable	The members of an interface are always declared as constant, their values are final
The class definition can contain the code for each of methods. That is the methods can be abstract or non-abstract.	The methods in an interface are abstract in nature. There is no code associated with them. It is later defined by the class implements the interface
It can be initiated by declaring objects	It cannot be used to declare objects. It can be inherited by a class
It can use various access specifiers like public, private or protected	It can only use the public access specifier.

3. What are packages? What are the benefits of defining packages?

Package is a mechanism for grouping a variety of classes and / or interfaces together.

The Grouping is usually done based on functionality.

Benefits:

- The classes contained in the packages of other programs can be reused.
- In packages, classes can be unique compared with classes in other packages.
- Packages provides a way to hide classes thus preventing other programs or packages from accessing classes that are meant for internal use only.

4. Which are the types of packages.

1. JAVA API Packages
2. User Defined packages

5. List Java API Packages.

Java API Packages:

A large number of classes grouped into different packages based on functionality.

1. **java.lang** :Language support classes. They include classes for primitive types,strings,math fnctions,threads and exceptions
2. **java.util** :Language utility classes such as vectors, hash tables, random numbers, date etc
3. **java.io** : Input/output support classes.
4. **java.awt** : Set of classes for implementing graphical user interface. They include classes for window, buttons,lists,menus and so on.
- 5.**java.net** : Classes for networking. They include classes for communicating with local computers as well as with internet servers
- 6.**java. applet** : Classes for creating and implementing applets

6. How we can hide the classes in package?

When we import a package using asterisk(*), all public classes are imported.

We may prefer to not import certain classes/we may like to hide classes from accessing from outside of the package. Such classes should not be declared public.

Ex:

```
package p1;
public class x    //public class
{
    body of x
}
class Y          //not public, hidden class
{
    body of y
}
```

7. What is thread?

A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a **thread**.

8. What is multithreading?

Multithreading is a conceptual programming concept where a program (process) is divided into two or more subprograms (process), which can be implemented at the same time in parallel.

9. What are advantages of multithreading?

- Enables programmers to do multiple things at one time.
- Programmers can divide a long program into threads and execute them in parallel which eventually increases the speed of the program execution
- Improved performance and concurrency
- Simultaneous access to multiple applications

10. Which are the different methods used for creating Thread?

A new thread can be created in 2 ways

1. By extending thread class
2. Implementing runnable interface (By converting class to thread)

11. Mention different thread exceptions.

- InterruptedException
- ThreadDeath
- InterruptedException
- IllegalArgumentException
- Exception

12. How do we set priorities for threads? Which are 3 priority constants?

We can set the priority of thread using the setPriority() method as follows:

```
ThreadName.setPriority(intNumber);
```

Thread class defines several priority constants:

- MIN_PRIORITY = 1
- NORM_PRIORITY = 5
- MAX_PRIORITY = 10

5 and 10 Marks Questions

1. Explain defining and implementing interface with example.

- **Defining interface**

Like classes, interfaces contain methods and variables but define only abstract methods and final fields.

Syntax:

```
interface interfacename
{
    variable declaration;
    methods declaration;
}
```

- **Implementing Interfaces**

```
class classname implements interfacename
{
    Body of classname
}
```

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 a;                                //interface object
        a=rect;                                //area refers to rect object
        System.out.println("Area of Rectangle="+a.compute(10,20));
        a=cir;                                //area refers to cir object
        System.out.println("Area of Circle="+a.compute(10,0));
    }
}

```

2. Write a program to implement multiple interfaces using interface.

```

class student //program in java to show multiple inheritance
{
    int rollNumber;
    void getNumber(int n)
    {
        rollNumber=n;
    }
    void printNumber()
    {
        System.out.println("RollNo is " +rollNumber);
    }
}
class test extends student
{
    float part1,part2;
    void getMarks(float a, float b)
    {
        part1=a;
        part2=b;
    }
    void putMarks()
    {
        System.out.println("Marks obtained");
        System.out.println("Marks Part1 "+part1);
        System.out.println("Marks Part2 "+part2);
    }
}

```

```

interface sports
{
float sportwt=6.0F;
void putwt();
}
class results extends test implements sports
{
float total;
public void putwt()
{
System.out.println("Sports Marks "+ sportwt);
}
void display()
{
total=part1+part2+sportwt;
System.out.println("Total marks= "+total);
}
}
class mainClass
{
public static void main(String srgs[])
{
results a=new results();
a.getNumber(10);
a.printNumber();
a.getMarks(10.0F,25.5F);
a.putMarks();
a.putwt();
a.display();
}
}

```

Output:

```

RollNo is 10
Marks obtained
Marks Part1 10.0
Marks Part2 25.5
Sports Marks 6.0
Total marks= 41.5

```

3. Explain creating, accessing and using packages with example.(10 Marks)

Creating Packages

- Creating a package in java is quite easy. Simply include a package command followed by name of the package as the first statement in java source file.
- This is the general form of the package statement:
package packagename;
- For example, the following statement creates a package called myPackage.
package myPackage;

Creating our own package involves the following steps

1. Declare the package at the beginning of a file using the form:

Package packagename;

2. Define the class that is to be put in the package and declare it public.
3. Create a subdirectory under the directory where the main source files are stored.
4. Store the listing as the classname.java file in the subdirectory created.
5. Compile the file. This creates .class file in the subdirectory.
6. Subdirectory name must match the package name exactly.

Accessing A Package

- Java system package can be accessed either using a fully qualified class name or using a shortcut approach through the import statement.

```
import package1 [.package2] [.package3].classname;
```

1. Fully Qualified class name:

Ex: pack.ClassA obj = new pack.ClassA();

2. import packagename.classname;

Ex: import java.awt.Color;

or

import packagename.*;

Ex: import java.awt.*;

Example:

ClassA.java

```
package p1;
public class ClassA
{
    public void displayA( )
    {
        System.out.println("Class A");
    }
}
```

testclass.java

```
import p1.*;
Class testclass
{
    public static void main(String str[])
    {
        ClassA obA=new ClassA();
        obA.displayA();
    }
}
```

4. Differentiate between multitasking and multithreading.

Multitasking	Multithreading
It is Process-based Multitasking	It is Thread-based Multitasking
The process is heavyweight.	Thread is lightweight.
Each process has its own address in memory i.e. each process allocates separate memory area.	Threads share the same address space.
Cost of communication between the process is high.	Cost of communication between the thread is low.
Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists etc.	No Switching

5. Explain extending thread class with example.

Extending thread class:

Make class runnable as thread by extending the class `java.lang.Thread`. This gives access to all thread methods directly. It includes following steps:

1. Declare the class as extending the Thread class.
2. Implement the `run()` method that is responsible for executing the sequence of code that the thread will execute.
3. Create a thread object and call the `start()` method to initiate the thread execution.

- **Declaring the class**

```
class MyThread extends Thread
{
    ....
}
```

- **Implementing the run() method**

We need to override run method in order to implement the code to be executed by our thread.

```
Public void run()
{
    .....
}
```

- **Starting New Thread**

```
MyThread th=new MyThread();
Th.start();           //invokes run() method
```


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.println("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[])
    {
        A obj1 = new A();
        B obj2 = new B();
        C obj3 = new C();
        obj1.start();  obj2.start();
        obj3.start();
    }
}
```

6. Explain life cycle of Thread with proper diagram.

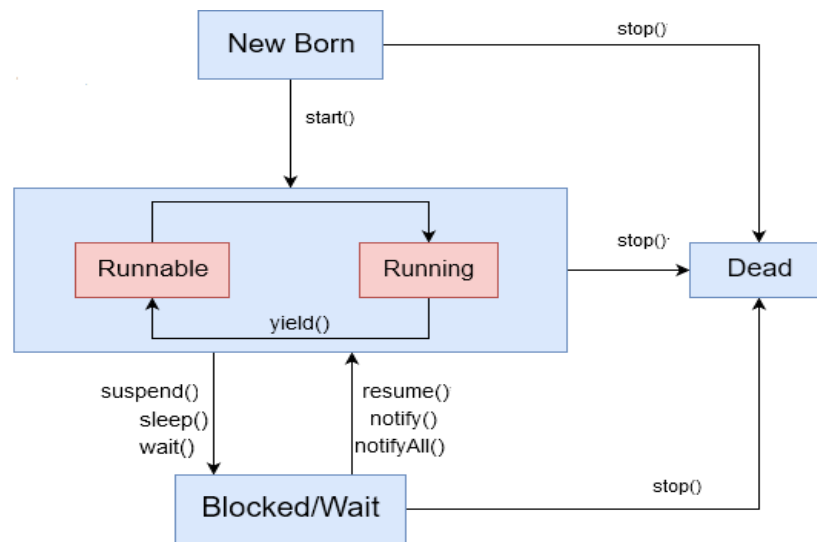


Fig : Life Cycle of Thread

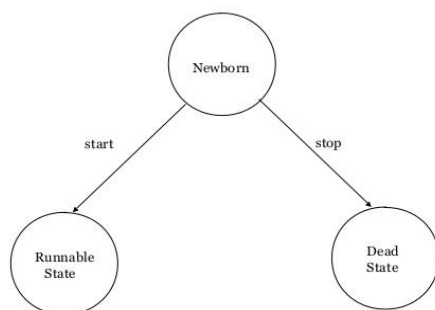
- There are different states of thread

1. Newborn state
 2. Runnable state
 3. Running state
 4. Blocked state
- Dead state

Newborn state

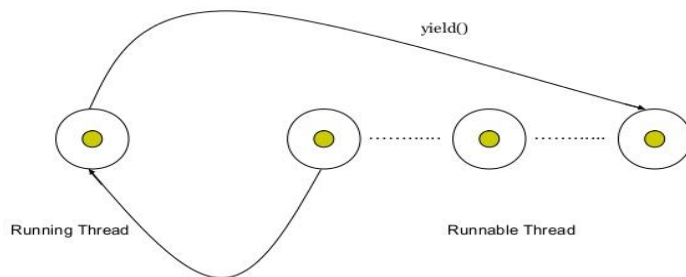
- When a thread object is created a new thread is born and said to be in newborn state.
- The Thread is not yet scheduled for running.
- It can do only following things
 - *Schedule it for running using start() method
 - *Kill it using stop() method

Scheduling a Newborn Thread



- **Runnable State:**
- If a thread is in this state it means that the thread is ready for execution and waiting for the availability of the processor.
- If all threads in queue are of same priority then they are given time slots for execution in round robin fashion i.e first come- first serve.
- If want to give control to another thread before its turn comes use `yield()` method.

Releasing Control Using `yield()`



Running State

It means that the processor has given its time to the thread for execution. A thread keeps running until the following conditions occurs

Thread give up its control on its own and it can happen in the following situations

- A thread gets suspended using **suspend()** method which can only be revived with **resume()** method
 - A thread is made to sleep for a specified period of time using **sleep(time)** method, where time in milliseconds
 - A thread is made to wait for some event to occur using **wait ()** method. In this case a thread can be scheduled to run again using **notify ()** method.
 - A thread is pre-empted by a higher priority thread
- **Blocked State:** If a thread is prevented from entering into runnable state and subsequently running state, then a thread is said to be in Blocked state. This happens when the thread is suspended, sleeping or waiting in order to satisfy certain requirements.
 - **Dead State:** A runnable thread enters the Dead or terminated state when it completes its task or otherwise terminates. We can kill thread by sending stop message to it at any state. A thread cab be killed as soon as it is born, or while running, or even when it is in blocked condition.

7. Explain with example different method of Thread.

Use of Yield() Method

Causes the currently running thread to yield to any other threads of the same priority that are waiting to be scheduled

Use of stop() Method

The stop() method kills the thread on execution

Use of sleep() Method

Causes the currently running thread to block for at least the specified number of milliseconds. You need to handle exception while using sleep() method.

Use of suspend() and resume() method

A suspended thread can be revived by using the resume() method. This approach is useful when we want to suspend a thread for some time due to certain reason but do not want to kill it.

Use of wait() and notify() Method

A thread is made to wait for some event to occur using **wait ()** method. In this case a thread can be scheduled to run again using **notify ()** method.

```
class A extends Thread
{
    public void run()
    {
        for(int i=1; i<=5; i++)
        {
            if(i==1) yield();
            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.println("From Thread B:j="+j);
            if(j==3) stop();
        }
        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);
            if(k==1)
            try
            {
                sleep(1000);
            }
            catch(Exception e)
            {
            }
        }
        System.out.println("Exit from C");
    }
}

class ThreadMethod
{
    public static void main(String args[])
    {
        A obj1 = new A();
        B obj2 = new B();
        C obj3 = new C();
        obj1.start();
        obj2.start();
        obj3.start();
    }
}

```

8. Write a program to demonstrate thread priority.

```

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.println("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[])
    {
        A obj1 = new A();
        B obj2 = new B();
        C obj3 = new C();
obj3.setPriority(Thread.MAX_PRIORITY);
obj2.setPriority(Tobj1.getPriority()+1);
obj1.setPriority(Thread.MIN_PRIORITY);
System.out.println("Start thread A");
obj1.start();
System.out.println("Start thread B");
obj2.start();
System.out.println("Start thread C");
obj3.start();
System.out.println("End of main thread");
    }
}

```

9. Explain with example implementing runnable interface.

The Runnable interface declares the run() method that is required for implementing threads in program.

To do this perform steps listed below:

1. Declare the class as implementing the Runnable interface.
2. Implement the run() method.
3. Create a thread by defining an object that is instantiated from this runnable class as the target of the thread.
4. Call the threads start() method to run the thread.

```

class X implements Runnable           //Step 1
{
    public void run()                  //step 2

```

```

{
    for(int i=1; i<=10; i++)
    {
        System.out.println("Thread X="+i);
    }
    System.out.println("End of THREAD X");
}
}
class RunnableTest
{
    public static void main(String args[])
    {
        X obj=new X();
        Thread th=new Thread(obj);           //step 3
        th.start();                           //step 4
        System.out.println("end of main thread");
    }
}

```

Java Programming

Unit-4

Managing Exceptions, Applet Programming:

2 marks questions

1. What is exception handling

The **Exception Handling in Java** is one of the powerful *mechanisms to handle the runtime errors* so that normal flow of the application can be maintained.

2. Which are the 2 types of errors?

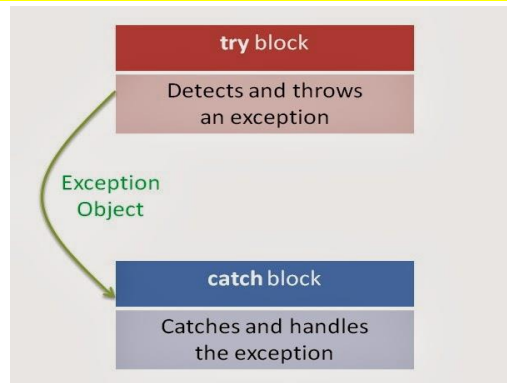
Compile time errors

Run time errors

3. List common Java Exception.

Name	Description
NullPointerException	Thrown when attempting to access an object with a reference variable whose current value is null
ArrayIndexOutOfBoundsException	Thrown when attempting to access an array with an invalid index value (either negative or beyond the length of the array)
IllegalArgumentException.	Thrown when a method receives an argument formatted differently than the method expects.
IllegalStateException	Thrown when the state of the environment doesn't match the operation being attempted,e.g., using a Scanner that's been closed.
NumberFormatException	Thrown when a method that converts a String to a number receives a String that it cannot convert.
ArithmeticException	Arithmetic error, such as divide-by-zero.

4. What is the basic concept of exception handling mechanism?



- Java uses a keyword try to preface 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 handles it appropriately
- The catch block is added immediately after the try block

5. How do we define try and catch block?

```
try
{
    statement; //generates an exception
}
catch(Exception-type e)
{
    statement; //processes the exception
}
```

- Java uses a keyword try to preface 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 handles it appropriately

6. What is finally block?

Java supports another statement known as finally statement that can be used to handle an exception that is not caught by any of the previous catch statements

Finally block can be used to handle any exception generated within a try block

It may be added immediately after the try block or after the last catch block shown as below

```
try
{
    .....
    .....
}
finally
{
    .....
}

try
{
    .....
}
catch
{
    .....
}
catch
{
    .....
}
```



```

.....
}
finally
{
.....
}

```

7. What is applet? What are the types of applet?

Applets are small Java programs that are primarily used in Internet computing.

1. Local Applet
2. Remote Applet

8. List the different state of applet life cycle.

- The applet states include:
 1. Born on initialization state
 2. Running state
 3. Idle state
 4. Dead or destroyed state
 5. Display state

9. How do applets differ from application programs?

Java Application	Applet
Java application contains a main method	An applet does not contain a main method
Does not require internet connection to execute	Requires internet connection to execute
Is stand alone application	Is a part of web page
Can be run without a browser	Requires a Java compatible browser
Uses stream I/O classes	Use GUI interface provided by AWT or Swings
Entry point is main method	Entry point is init method
Generally used for console programs	Generally used for GUI interfaces

10. List the attributes of Applet tag.

CODE=AppletFileName.class→Specifies the names of the applet class to be loaded. That is the name of the already compiled .class file

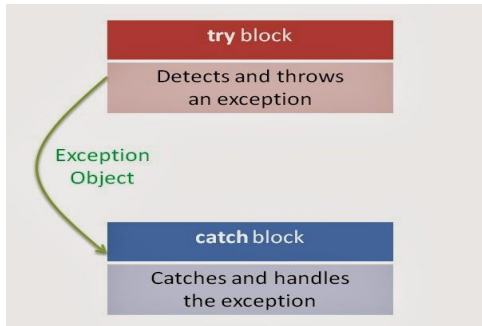
WIDTH=pixels→These attributes specify the

HEIGHT=pixels→ width and height of the space on the HTML page that will be reserved for the applet

HSPACE=pixels(optional)→Used only when ALIGN is set to LEFT or RIGHT

5 and 10 Marks questions

1. Explain exception handling using try catch finally block with example.



- Java uses a keyword try to preface 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 handles it appropriately

```
try
{
    statement; //generates an exception
}
catch(Exception-type e)
{
    statement; //processes the exception
}
```

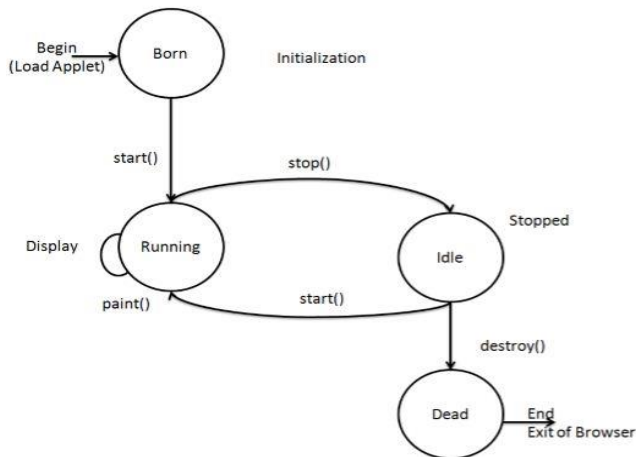
The catch block is added immediately after the try block

- Java supports another statement known as finally statement that can be used to handle an exception that is not caught by any of the previous catch statements
- When a finally block is defined this is guaranteed to execute, regardless of whether or not an exception is thrown

```
public class exp
{
    public static void main(String args[])
    {
        int a=10,b=0,c;
        try
        {
            c=a/b;
            System.out.println(c);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Devide by zero");
        }
        finally
        {
            System.out.println("This Block always Executes...!");
        }
    }
}
```

```
}  
}
```

2. Explain applet life cycle.



Initialization state:

Applet enters this state when it is first loaded. This is achieved by calling `init()` method. The `init()` method is the first method to be called. Initialize variables in this method. This method is called only once during the life time of your applet.

Running state:

Applet enters into running state when it calls `start()` method. The `start()` method is called after `init()`. It is also called to restart an applet after it has been stopped. Where as `init()` is called only once, `start()` method may be called any number of times. When a user leaves a web page and comes back, the applet resumes.

Idle or stopped state:

Applet becomes idle when it is stopped. Stopping automatically occurs when we leave the page containing the current applet. We can achieve this by calling `stop()` method.

`stop()` method suspends applet execution until the user clicks on it.

Dead state :

Applet is said to be dead when it is removed from memory. This occurs by calling `destroy()` method. The `destroy()` method performs shutdown activities. It also removes the applet from the memory.

Display state :

Applet moves to the display state whenever it has to perform some output operation on the screen. The `paint()` method is called to accomplish this task. `Paint()` method does absolutely nothing. We have to override this method to display the required graphics.

Public void `paint(Graphics g)`

```
{  
    .....  
}
```

3. Describe the various section of web page.

1. Comment section(optional)
2. Head section(optional)
3. Body section

Comment section :

This element is used to add a **comment** to an **HTML** document. An **HTML comment** begins with `<!--` and the **comment** closes with `-->`. **HTML comments** are visible to anyone that views the page source code, but are not rendered when the **HTML** document is rendered by a browser.

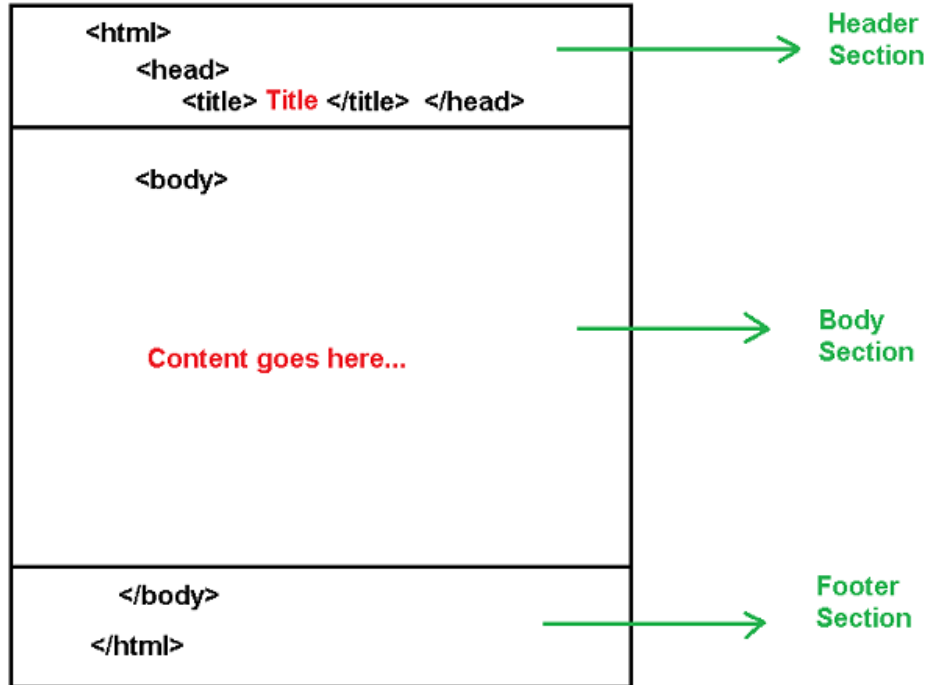
Head section:

The `<head>` tag is what actually begins the **head section** of an **HTML** document. The **head section** of an **HTML** document contains several tags that specify important information about the webpage such as title,

keywords and descriptions (for search engines), location of style sheets, scripts, and more.

Body section:

HTML body section is a main contain **section** of web page all contain that will be seen when the user loads the webpage. **HTML body section** supported all the contains such as text, hyper-links, images, Special Character, lists, tables, frames, forms etc.



4. List HTML tags and its functions.

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h <i>n</i> > ... </h <i>n</i> >	Delimits a level <i>n</i> heading
 ... 	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
 ... 	Brackets an unordered (bulleted) list
 ... 	Brackets a numbered list
 ... 	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
	Displays an image here
 ... 	Defines a hyperlink

5. Explain passing parameters to Applets with example.

We can supply user-defined parameters to an applet using <PARAM...>tags.Each <PARAM...> tag has a name attribute such as color and a value attribute such as red

Inside the applet code, the applet can refer to that parameter by name to find its value

For eg,we can change colour of the text displayed to red by an applet as follows

```
<APPLET>
```

```
<PARAM=color VALUE="red">
```

```
</APPLET>
```

- 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 init() method in the applet to get hold of the parameters defined in the <PARAM...> tags.This is done using the getParameter() method,which takes one string argument representing the name of the parameter and returns a string containing value of that parameter

Example:

```
import java.applet.Applet;  
import java.awt.Graphics;  
public class UseParam extends Applet{  
    public void paint(Graphics g){  
        String str=getParameter("msg");  
        g.drawString(str,50, 50);  
    }  
}
```

myapplet.html

```
<html>
<body>
<applet code="UseParam.class" width="300" height="300">
<param name="msg" value="Welcome to applet">
</applet>
</body>
</html>
```

6. Write a program to get the input from the user and display in applet area.

```
import java.awt.*;
import java.applet.*;
public class UserIn extends Applet
{
    TextField text1,text2;
    text1=new TextField(8);
    text2=new TextField(8);
    add (text1);
    add (text2);
    text1.setText ("0");
    text2.setText ("0");
}
public void paint(Graphics g)
{
    int x=0,y=0,z=0;
    String s1,s2,str;
    g.drawString("Enter the number in each box",10,50);
    try
    {
        s1=text1.getText();
        x=Integer.parseInt(s1);
        s2=text2.getText();
        y=Integer.parseInt(s2);
    }
    catch(Exception ex)
    {
    }
    z=x+y;
    s=String.valueOf (z);
    g.drawString("Sum is",10,15);
    g.drawString(s, 100, 75);
}
Public Boolean action (Event event , Object object)
{
    repaint ( );
    return true;
}
}
```

HTML Code:

```
<html>
  <applet
    code=UserIn.class
    width=300
    height=200>
  </applet>
</html>
```

Unit-5**Graphics Programming, Input / Output: Graphics Programming****2 Marks questions**

1. List the drawing methods of graphics class.

Method	Description
clearRect ()	Erases a rectangular area of the canvas
copyArea ()	Copies a rectangular area of the canvas to another area
drawArc ()	Draws a hollow arc
drawLine ()	Draws a straight line
drawPolygon ()	Draws a hollow polygon
fillArc ()	Draws a filled arc
setColor ()	Sets the drawing color
setFont ()	Sets the font
fillOval ()	Draws a filled oval

2. Explain drawing lines and rectangles methods from graphics class.

The drawLine () method takes two pair of coordinates, (x1, y1) and (x2, y2) as arguments and draws a line between them

Ex: g.drawLine (10,10, 50, 50);

drawRect() method. This method takes four arguments. The first two represent the x and y coordinates of the top-left corner of the rectangle, and the remaining two represent the width and the height of the rectangle

Ex: g.drawRect (50, 80, 150, 100)

3. Explain drawOval and fillOval() methods from graphics class

- The drawOval() method takes four arguments: the first two represent the top-left corner of the imaginary rectangle and the other two represent the width and height of oval itself
- The drawOval() method draws outline of an oval, and the fillOval() method draws a solid oval.

```
g.drawOval (20, 20, 200, 120);  
    g.setColor (Color.green);  
    g.fillOval (70, 30, 100, 100); //This is a circle
```

4. Explain the purpose of each argument used in the drawArc() method.

The drawArc() designed to draw arcs takes six arguments. The first four are the same as the arguments for drawOval() method and the last two represent the starting angle of the arc and the number of degrees(sweep angle) around the arc.

```
g.fillArc(60, 125, 80, 40, 180, 180);  
g.drawlArc(60, 125, 80, 40, 180, 180);
```

5. Which are the three different ways of drawing polygon?

We can draw polygons using drawPolygon() method of Graphics class. This method takes three arguments

- An array of integers containing x coordinates
- An array of integers containing y coordinates
- An integer for total number of points

6. What is stream? How is the concept of streams used in Java?

A stream presents a uniform, easy-to-use, object-oriented interface between the program and the input/output devices.

7. What are input and output streams?

Input stream classes that are used to read 8-bit bytes include a super class known as InputStream and a number of subclasses for supporting various input-related functions

- Output stream classes are derived from the base class OutputStream.
- Like InputStream, the OutputStream is an abstract class and therefore we cannot instantiate it
- Several subclasses of the OutputStream can be used for performing the output operations

5 and 10 marks questions.

1. Write a program to demonstrate different drawing methods of graphics class.

```
import java.awt.*;  
import java.applet.*;  
public class test extends Applet  
{  
    public void paint(Graphics g)  
    {  
        g.setColor(Color.red);  
        g.drawRect(80, 80, 80, 50);  
        g.drawOval(40, 40, 50, 50);  
        g.fillOval(60, 60, 50, 50);  
        g.drawLine(10,10,45,50);  
        g.fillArc(60,125,80,40,180,180);  
        g.drawRoundRect(10,1000,80,50,10,10);  
    }  
}
```


HTML Code:

```
<html>
<body>
<applet
Code=test.class
Width=400
Height=400>
</applet>
</body>
</html>
```

2. Explain line graphs with example.

We can design applets to draw line graphs to illustrate graphically the relationship between two variables. Consider the table of values

X	0	60	120	180	240	300	360	400
Y	400	280	220	140	60	60	100	220

The variation of Y , when X is increased can be shown graphically using the following program

```
Import java.awt.*;
Import java.applet.*;
Public class TableGraph extends Applet
{
    int x [ ] = {0, 60,120, 180, 240, 300,360, 400};
    int y [ ]= { 400, 280, 220, 140, 60, 60, 100, 220};
    int n = x.length;
    public void paint (Graphics g)
    {
        g.drawpolygon (x, y, n);
    }
}
```

3. Explain drawing bar charts with example.

- Applets can be designed to display bar charts, which are commonly used in comparative analysis of data.
- Consider the table below

Year	1991	1992	1993	1994
Turnover (Rs crores)	110	150	100	170

- These values may be placed in an HTML file as PARAM attributes and then used in an applet for displaying a bar chart

```

import java.awt.*;
import java.applet.*;
public class BarChart extends Applet
{
    int n=0;
    String label[];
    int value[];
    public void init() {
        setBackground(Color.pink);
        try {
            int n = Integer.parseInt(getParameter("Columns"));
            label = new String[n];
            value = new int[n];
            label[0] = getParameter("label1");
            label[1] = getParameter("label2");
            label[2] = getParameter("label3");
            label[3] = getParameter("label4");
            value[0] = Integer.parseInt(getParameter("c1"));
            value[1] = Integer.parseInt(getParameter("c2"));
            value[2] = Integer.parseInt(getParameter("c3"));
            value[3] = Integer.parseInt(getParameter("c4"));
        }
        catch(NumberFormatException e){ }
    }
    public void paint(Graphics g)
    {
        for(int i=0;i<4;i++) {
            g.setColor(Color.black);
            g.drawString(label[i],20,i*50+30);
            g.setColor(Color.red);
            g.fillRect(50,i*50+10,value[i],40);
        }
    }
}

```

```

<html
<applet code=BarChart .class
    width=300 height=250>
    <param name = "columns" value = "4">
    <param name=c1 value=110>
    <param name=c2 value=150>
    <param name=c3 value=100>
    <param name=c4 value=170>
    <param name=label1 value=1991>
    <param name=label2 value=1992>
    <param name=label3 value=1993>
    <param name=label4 value=1994>
    <param name=Columns value=4>
</applet>
</html>

```