Important Question for Java in GTU

1. List out Feature of Java and Explain any three feature.
2. Explain JVM.
3. Explain Structure of Java.
4. Main Method of java.
5. What is constructor? Explain any one with Example.
6. Explain Parameterize Constructor with example
7. Explain Copy Constructor with example
8. Explain Method Overloading with example.
9. Explain Method Overloading VS Method Overriding
10. Explain Wrapper Class With example
11. Explain Garbage Collection.
12. Give the Difference between String Class and String Class Buffer
13. Explain any three methods of String Class and String Buffer Class.
14. Explain Type Casting and Type Conversion
15. Explain OOP's Concept
16. Explain Inheritance With example
17. Explain Interface With example
18. List out System Define Package
19. Explain Life Cycle of Thread
20. Write a program to Create and Read text file
21. Write a program to Create and Write text file
22. Explain:-
    - static
    - final
    - super
    - this
    - Throw, Throws
23. Explain Method Overriding with example.
24. Define Package and write a step to create Package.
25. Explain Exception Handling.
26. Explain Types of Errors in Java.
27. Write a program for generating Exception "Divided by Zero".
28. Explain abstract keyword with example.
29. Explain Dynamic method dispatch with example.

1. **List out Feature of Java and Explain any three feature.**

Ans:

1. **C**ompiled and interpreted
2. **P**latform independent
3. **P**ortable
4. **O**bject oriented
5. **R**obust and secure
6. **D**istributed
7. **F**amiliar, simple and small
8. **M**ultithreaded and interactive
9. **H**igh performance
10. **D**ynamic and extensible

Compiled and interpreted

- ✓ Computer language is either compiled or interpreted. Java combines both so it is two stage system
  - ✓ First java compiler translates source code into byte code.
  - ✓ Java interpreter generates machine code that can be directly executing by machine that is running the program.

Platform independent

- ✓ When java program is compiled by java compiler it will produces on object file which contains bytecode. This byte codes are not machine/CPU specific, it can be interpreted by JVM (Java Virtual Machine).so that code can be run on any JVM of computer system.
- ✓ The Same byte codes can be executed by any JVM on any platform.
- ✓ Thus java is platform independent.

Portable

- ✓ Java's important feature is portability.
- ✓ Java program can be easily moved from one computer system to another, anywhere and anytime.
- ✓ Java provides Write Once Run Anywhere (WORA), which makes the java language portable.

Object oriented

- ✓ Java is true (pure) object oriented language because object is the outer most level of data structure in java.
- ✓ Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

2. Explain JVM.

Ans:

- ✓ All language compilers translate source code into machine code for a specific computer.
- ✓ Java is platform independent or machine neutral!!!
- ✓ Java compiler produces an intermediate code (byte code or virtual machine code) for a machine that does not exist. That machine is Java Virtual Machine (JVM).
- ✓ JVM exists only inside the computer memory.
- ✓ JVM is a simulated computer within the computer and does all major functions of real computer.

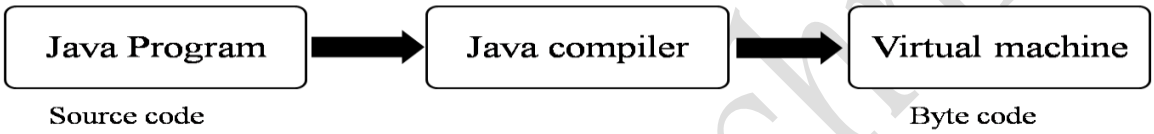


Figure :4.1 **Process of compilation**

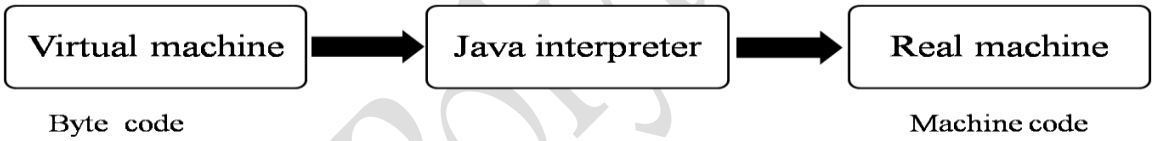
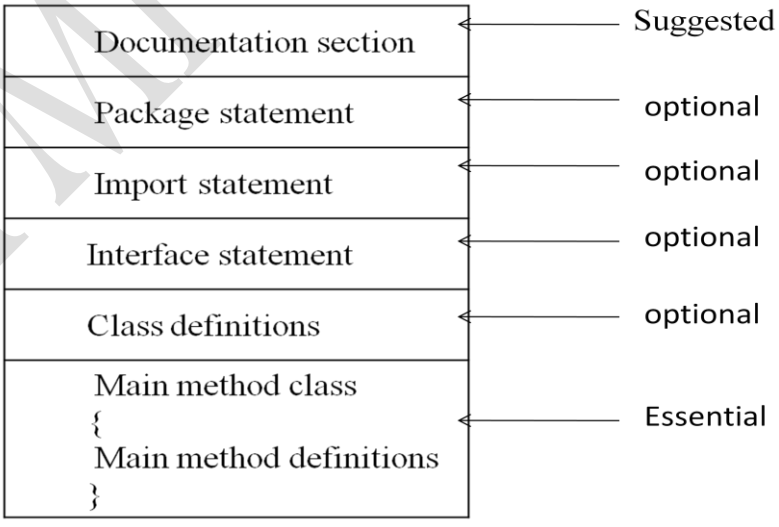


Figure :4.2 **Process of converting byte code into machine code**

3. Explain Structure of Java.



Basic structure of java program

Documentation section:

- This section contains a set of comment lines Like the name of the program, author and other details.
- Java supports three types of comment line:

Single comment line	:	//
Multiline	:	/* ……. */
Documentation line	:	/** ……. */

Package statement:

- The statement declares a package name.
- package student ; //student is package
- package statement is optional.

Import statements:

- This is similar to the #include statement in C.
- import packagename.classname;
- Using import statement ,we can access to classes that are part of

Interface statement:

- Interface is like a class but includes a group of method declarations.
- This is also an optional section.
- Java does not support multiple inheritances, so java use interface to implement multiple inheritances.

Class definitions:

- A java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program.

Main method class:

- Every java stand alone program requires a main method as its starting point, this class is essential part of a java program.
- Simple java program may contain only this part.

4. Main Method of java.

Ans:

```
class Test
{
public static void main(String args[ ])
{
        System.out.println( "welcome to Java");
}
```

```
}
```

public: The keyword public is an access modifier. It declares main method as unprotected .it makes method accessible to all other classes.

Static: It is a keyword, which declares method is for entire class and not a part of any object of class.

Void does not return any value.

String args declares parameter names args which contain an array of object of class type string.

5. What is Constructor? Explain any one with Example.

Ans:

- Same name as class name is known as Constructor.
- Constructor is a special Type of method.
- Constructor does not have any return type not even Void.
- There are three Type of Constructor
  1. Default Constructor
  2. Parameterize Constructor
  3. Copy Constructor
- Default Constructor

  When user does not define any type of Constructor then Java create System define constructor which is known as Default Constructor.

  Example:

```
class A
{
                A( )
               {
                System.out.println("This is Default Constructor");
                }
                void display( )
                {
                System.out.println("This is Simple Method");
                }
}
class Testc
{
```

```
        public static void main(String args[ ])
        {
                A a1=new A( );
                A1.display( );
        }
}
```

5. Explain Parameterized Constructor With example

Ans:

- Same name as class name is known as Constructor.
- Constructor is a special Type of method.
- Constructor does not have any return type not even Void.
- There are three Type of Constructor
  1. Default Constructor
  2. Parameterize Constructor
  3. Copy Constructor
- When we pass argument in Constructor then it is known as Parameterized Constructor.

Example:

```
class A
{
        A( int a)
        {
        System.out.println("The value of a is "+i);
        }
        void display( )
        {
        System.out.println("This is Simple Method");
        }
}
class Test
{
        public static void main(String args[ ])
        {
                A a1=new A( 10);
                A1.display( );
        }
```

```
}
```

6. Explain Copy Constructor with example.

Ans:

- Same name as class name is known as Constructor.
- Constructor is a special Type of method.
- Constructor does not have any return type not even Void.
- There are three Type of Constructor
  1. Default Constructor
  2. Parameterize Constructor
  3. Copy Constructor
- Copy Constructor copy the state (date) of one object to another object.
- **Example:**

```
class A
{
   int x;
   A(int a)
   {
   x=a;
   }
   A(A obj)
   {
   System.out.println("copy cosnstructor invoked");
   System.out.println("x="+obj.x);
   }
}
class Copy1
{
public static void main(String args[])
{
            A a2=new A(10);
            A a3=new A(a2);
}
}
```

7. Explain Parameterized Constructor With example

Ans:

- Same name as class name is known as Constructor.
- Constructor is a special Type of method.
- Constructor does not have any return type not even Void.
- There are three Type of Constructor
  1. Default Constructor
  2. Parameterize Constructor
  3. Copy Constructor
- When we pass argument in Constructor then it is known as Parameterized Constructor.

Example:

```
class A
{
        A( int a)
      {
        System.out.println("The value of a is "+i);
        }
        void display( )
        {
        System.out.println("This is Simple Method");
        }
}
class Test
{
    public static void main(String args[ ])
    {
            A a1=new A( 10);
            A1.display( );
    }
}
```

8. Explain Copy Constructor with example.

Ans:

- Same name as class name is known as Constructor.
- Constructor is a special Type of method.
- Constructor does not have any return type not even Void.
- There are three Type of Constructor
  1. Default Constructor
  2. Parameterize Constructor
  3. Copy Constructor
- Copy Constructor copy the state (date) of one object to another object.
- **Example:**

```
class A
{
   int x;
   A(int a)
   {
   x=a;
   }
   A(A obj)
   {
   System.out.println("copy cosnstructor invoked");
   System.out.println("x="+obj.x);
   }
}
class Copy1
{
public static void main(String args[])
{
            A a2=new A(10);
            A a3=new A(a2);
}
}
```

9. Explain Method Overloading VS Method Overriding.

Ans:

Method Overloading	Method Overriding
When same method name but different parameters and return type is used in a single class then it is known as Method Overloading.	When same method name, same argument and same return type is used in a sub class and superclass is called Method Overriding.
Method Overloading can be done in same class.	Method Overriding can be done in different class.
In a Method Overloading parameter must be different.	In a Method Overriding parameter must be same.
Method Overloading is the example of compile time Polymorphism.	Method overriding is the example of run time polymorphism.
Return type can be same or different in method Overloading.	Return type must be same in Method overriding.
For implementing method overloading we do not require inheritance.	For implementing method overloading we must require inheritance.

10. Explain Wrapper Class with example.

Ans:

- Wrapper class is used to convert any datatype in to object.
- The primitive datatypes are not object. They do not belong to any class. They are defining in the language itself.
- Sometimes, it is required to convert datatype into object in java.
- For example: – In a java Vector class does not support primitive datatype. So, we have to compulsory convert primitive datatype into object.
- Wrapper class solves converting simple type.

Primitive Datatype	Wrapper Class
boolean	Boolean
char	Character
double	Double

float	Float
int	Integer
long	Long

- Example:

```
class WrapperDemo
{
        public static void main(String args[ ])
        {
                int a=10;
                Integer i=Integer.valueOf(a);
                System.out.println("a="+a+"i="+i);
        }
}
```

11. Explain Garbage Collection.

Ans:

- Java does not support Destructor to free up the memory. For that java support Garbage Collection.
- A Java runtime environment delete object periodically when it determine that they are no longer being used. This process is known as Garbage Collection.
- In a java Garbage Collection process is done by Garbage Collector () and Finalizer () for memory relocation.
- Java Garbage Collection is the process by which java program perform automatic memory management.

12. Give the Difference between String Class and String Class Buffer.

Ans:

String class	StringBuffer class
It creates fixed length string.	It created strings of flexible(variable) length
It can't be changed.	It can modify in terms of length and content.
We can't insert character and substrings in the middle string.	We can insert characters and substrings in the middle of a string or append another string to the end.

String class methods are	StringBuffer class methods are
Concat( )	setCharAt( )
Equals( )	append( )
Replace( ) etc.	insert( ) etc.

13. Explain any three methods of String Class and String Buffer Class.

Ans:

String Class:

1) **Concat () Method:-**

- ✓ The concat () method is used to join one string with another string.
- ✓ Concat () concatenate (join) atr1 and str2.

Example:

```
String str1="Bahubali1";
String str2="Bahubali2";
String str3=str1.concat(str2);
```

2) **Equals() Method:-**

- ✓ Equals method is used to compare two strings.
- ✓ It compares a content of two string and return the Boolean value.
- ✓ If both string are same then return true else return false
- ✓ Equals() method compare actual contain of two string object.

Syntax:

String1.equals(string2);

Example:s

```
String str1="Hello";
String str2="hello";
String str3="Hello";
Str1.equals(str2);                    //returns false
Str1.equals(str3);                    //returns true
```

3) **Replace() method**

- ✓ The replace () is used to replace all occurrence of the specified character with given character.

Example:

```
Str1="Patel";
Str1.replace ("p","f");
```

Output: "fatel"

String Buffer Class:

1) **setCharAt(n,x) Method:-**
   - ✓ it modifies the n^{th} character of x.

   Example:

   String str1="VXMP";

   Str1. **setCharAt (1,'P');**

   Output: VPMP

2) **append() Method:-**
   - ✓ Join or append the string str2 at the end of str1.

   Syntax:

   String1.append (string2);

   Example:

   Str1="vpmp";

   Str2="polytechnic";

   Str1.append (str2);

   Output: "vpmp polytechnic"

3) **Insert () method:-**
   - ✓ It inserts the string str2 at the position n of the string str1.

   Syntax:

   String1.insert (n, string2);

   Example:

   Str1="16ce1 class";

   Str2="Diamond";

   Str1.insert (6, str2);

   Output: "16ce1 Diamondclass"

14. Explain Type Casting and Type Conversion

Ans:

- The process of converting one datatype into another datatype is called type casting.
- It is used when there is a need to store a value of one type into a variable of another type.
- **Syntax:**

  Type variable_name1= (type) variable_name2;

- **Example:**

```
int m=5;
byte n= (byte) m; //convert int into byte m.
```

- **Automatic casting:**

It is possible to assign value of one type to a other type without casting is known as automatic type conversion.

There are two operations available in automatic casting.

1. Winding
2. Narrowing

1. Winding:

- ✓ Process of assigning smaller to larger type is known as winding or promotion.
- ✓ Example:

```
byte b=5;
int a=b;
```

2. Narrowing:

- ✓ Process of assigning large type to smaller type is known as Narrowing.
- ✓ Narrowing ay result in lost of information.
- ✓ Example:

```
float c=25.43f;
int a=(int)c;
```

15. Explain OOP's Concept.

Ans:

1. Object and Class
2. Data abstraction and Data Encapsulation
3. Inheritance
4. Polymorphism
5. Data Binding
6. Message Passing

1. Object and Class:

- ✓ Object is a runtime entity in object oriented system.
- ✓ Object may represent a person, place or any item.
- ✓ **Class:** class is collection of logical related data items which include variables and method for data.

2. Data abstraction and Data Encapsulation:

- ✓ Data abstraction refer as representing require features without including details or explanation.
- ✓ Data Encapsulation means wrapping of data and methods into a single unit.

3. Inheritance:

- ✓ Inheritance is a process by which object of one class used the properties of another class.
- ✓ It provides reusability of code.

4. **Polymorphism:**

- ✓ Polymorphism means the ability to take more than one form that means the same operation have behaved on different classes.

5. Data Binding:

- ✓ Binding refers to the linking of procedure call to the code to be executed in response to call.

6. Message Passing:

- ✓ Message communication done by sending and receiving information from set of object.

16. What is Inheritance? Explain any one types of inheritance with example.

Ans:

- Inheritance is a process by which object of one class run the properties of object of another class.
- Inheritance provides reusability of code.
- Reusing the properties of existing class is called inheritance.
- **Types of Inheritance**
  1. Single Inheritance
  2. Hierarchical Inheritance
  3. Multilevel Inheritance
  4. Multiple Inheritances
  5. Hybrid Inheritance

  Note: java does not support multiple inheritance and hybrid inheritance.

1. Single Inheritance

- ✓ Only one super class.
- ✓ Every class has one and only one super class.
- ✓ If class Inherit only single class then it is known as single inheritance.

- ✓ **Syntax:**

```
class a
{
        //code
}
class b extends a
{
        //code
}
```

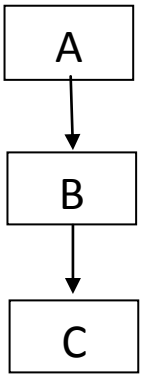
- ✓ **Example:**

```
class a
{
    int i;
}

class b extends a
{
    Void display()
    {
        System.out.println("the value of I is ="+i);
    }
}
class singledemo
{
    public static void main(String args[])
    {
        b b1=new b();
        b1.i=7;
        b1.display();
    }
}
```

2. Multilevel Inheritance:

- ✓ New class is derived from derive class.
- ✓ Java uses this type of inheritance mainly implementing its class library.



Syntax:

```
class a
{
        //statement
}
class b extends a
{
        //statement
}
class c extends b
{
        //statement
}
```

Example:

```
class a
{
    int I;
}
class b extends a
{
    int j;
    void display()
    {
```

```
            System.out.println("I in class b"+j);
        }
}
class c extends b
{
        void sum()
        {
                System.out.println("the sum is "+(i+j));
        }
}
class MultilevelDemo
{
        public static void main(String args[])
        {
                b b1=new b();
                b1.i=5;
                c c1=new c();
                c1.display();
                c1.i=10;
                c1.j=20;
                c1.sum();
        }
}
```

17. Explain Interface with example.

Ans:

- Java does not support multiple inheritances.
- Multiple inheritance mean one sub class and more than one super class.
- Java provides alternate solution for multiple inheritances that is interface.
- Using interface we can implements multiple inheritance.
- Interface is basically a kind of class.
- Interface contains methods and variable but it defines only abstract method and final variable.
- Interface does not specify any code to implement methods (method does not have body).
- Java use interface keyword to create interface.

- **Syntax:**

```
interface interfaceName
{
        Variable Declaration;
        Method Declaration;
}
```

- **Example:**

```
interface A
{
    int i=10;
}
interface B
{
    int j=20;
}
class C implements A,B
{
    void sum()
    {
        System.out.println("the sum is "+(i+j));
     }
}
class InterFaceDemo
{
    public static void main(String args[])
    {
        C c1=new C();
        c1.sum();
    }
}
```

18. List out System Define Package.

Ans:

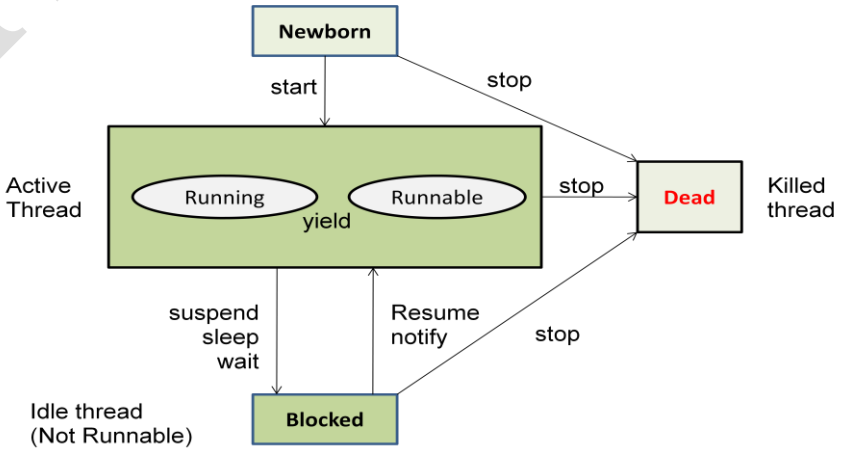
Package name	Description
Language support package **(lang)**	It includes classes and methods required for implementing basic feature of Java.
Utility package **(util)**	Collection of classes to provide utility functions such as date and time functions.
Input output package **(io)**	A collection of classes required for input/output manipulation.
Networking package **(net)**	Collection of classes for communicating with other computers via internet.
Abstract Window toolkit package **(awt)**	Collection of classes that implements Platform independent Graphical User Interface
Applet package **(applet)**	Collection of classes that allows us to create java applets.

19. Explain Life Cycle of Thread

Ans:

There are five states in Thread life cycle.

1. **Newborn state**
2. **Runnable state**
3. **Running state**
4. **Blocked state**
5. **Dead state**



1) New born state

- ✓ When we create a thread object, the thread is born and is said to be in newborn state.
- ✓ The thread is not still scheduled for running.

2) Runnable state

- ✓ The next state after the newborn state is Runnable state.
- ✓ Runnable state means that the thread is ready for execution and is waiting for the availability of the processor.
- ✓ The threads has joined waiting queue for execution based on priority of thread.

3) Running state

- ✓ Running means that processor has given its time to the thread for its execution.
- ✓ A running thread may change its state to another state using resume (), modify (), sleep (), wait () methods etc.

4) Blocked state

- ✓ A thread is said to be blocked when it is prevented from entering into the runnable state and subsequently the running state.
- ✓ This happens when the thread is suspended, sleeping or waiting in order to satisfy certain requirements.

5) Dead state

- ✓ Every thread has a life cycle. A running thread ends its life when it has completed executing its run ( ) method.
- ✓ It is natural death. However we can kill it by sending the stop message to it as any state thus causing a premature death for it.

20. Write a program to create and Read text file.

Ans:

```
import java.io.*;
public class ReadingByte
{
        public static void main(String args[])
        {
                FileInputStream f1 = null;
```

```
                int b;
                try
                {
                        F1 = new FileInputStream ("in.txt");
                        While ((b = f1.read ())! = -1)
                        {
                                System.out.println ((char) b);
                        }
                        f1.close ();
                }
                catch(IOException e)
                {
                System.out.println("Sorry..!! File Not Found...!!!");
                }
        }
}
```

Output :

I LOVE INDIA

21. Write a program to Create and Write text file.

Ans:

```
import java.io.*;
public class WriteByte
{
        public static void main(String args[])
        {
        File f1=new File ("input.txt");          \\ to create new file
        FileOutputStream outfile = null;
        byte cities[] = {'I','L','O','V','E','J','A','V','A'};
                try
                {
                        outfile = new FileOutputStream (f1);
                        outfile.write (cities);
```

```
                }
                catch (IOException e)
                {
                                System.out.println(e);
                                 System.exit(-1);
                }
                System.out.println("Write Byte");
                System.out.println("Thank You...!!!");
        }
}
```

Output :

Write Byte

Thank You...!!!

22. Explain following

a) **Static**

b) **Final**

c) **Super**

d) **This**

e) **Throw, Throws**

Ans:

A) static:

- ✓ Static is a member that is common to all object and Exide without using any object.
- ✓ Static is a keyword.
- ✓ Static method and variable are called using only with class name.

Syntax:

Classname.variablename;

Classname.methodname ( );

- ✓ We can define static variable, static method and static block.

1. **Static variable** — static int count;
2. **Static method** — static int max(int x, int y)

```
{
        //code
}
```

3. static block

```
static
{
    //code
}
```

Example:

```
class staticDemo
{
    static int mul(int x,int y)
    {
        return(x*y);
    }
    static int add(int x,int y)
    {
        return(x+y);
    }
}
class test
{
    public static void main(String args[])
    {
        int a=StaticDemo.mul(10,20);
        int b=staticDemo.add(a,40);
        System.out.println("the value of b is"+b);
    }
}
```

B) Final:

- ✓ final is a keyword.
- ✓ Final means cannot be change or it is final (fix).
- ✓ It is a same us constant in c++.
- ✓ We can use final keyword in java with variable, class, method.

Final variable:

- ✓ When we declare variable as final then we cannot change the value of variable..
- ✓ Final variable name must be declared in capital letter.
- ✓ All final variables must have initial value.

Syntax:

Final variable_name=value;

Example:

Final float pi=3.14;

Final class:

- ✓ If we declare class with final keyword then this class can't be inherit.
- ✓ If we declare class as final. It prevents any unwanted extension to the class.
- ✓ Many classes of java.lang package are declaring as a final class.

```
final class a
{
        //statements
}
class b extends a        //invalid it gives error.
{
        //statements
}
```

Final Method:

- ✓ When we declare method as final then we cannot overwrite this method in subclass.

```
class a
{
        final void display ()
        {
                //statements
        }
}
class b extends a
{
        void display() //invalid
        {
                //statements
        }
}
```

C) Super

- ✓ Super is a keyword.
- ✓ Constructor which is defining in a subclass is called sub class constructor.
- ✓ 'Super' keyword is used to invoke constructor method of super class.
- ✓ Super keyword is used to access variable of super class which have same name in sub class.

Conditions to used 'super' keyword: -

- ✓ Super keyword only defines in subclass constructor method.
- ✓ In a subclass constructor method we have to first mention the super keyword statement.
- ✓ The parameter in a super call must be match with the declare parameter in super class.

Example:

```
class Animal
{
String color="white";
}
class Dog extends Animal
{
String color="black";
void printColor()
{
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1
{
public static void main(String args[])
{
Dog d=new Dog();
d.printColor();
}
}
```

D) this

- ✓ this keyword is a reference variable that refers to the current object.
- ✓ this can be used to refer current class instance variable.
- ✓ this can be used to invoke current class method.
- ✓ this() can be used to invoke current class constructor.
- ✓ **Example:**

```
class S
{
int a,b;
Student(int a, int b)
{
this.a=a;
this.b=b;
}
void display()
{
System.out.println(a+" "+b+" ");
}
}
 class ThisDemo
{
public static void main(String args[])
{
S s1=new S(10,20,10);
s1.display();
}
}
```

E) Throw and Throws

Throw

- ✓ The Java throw keyword is used to explicitly throw an exception.
- ✓ We can throw either checked or unchecked exceptions in java by throw keyword.
- ✓ The throw keyword is mainly used to throw custom exception.

Syntax:

throw exception;

Example:

```
public class TestThrow1
{
        static void validate(int age)
        {
        if(age<18)
        {
                throw new ArithmeticException("not valid");
        }
        else
        {
                System.out.println("welcome to vote");
        }
        }
        public static void main(String args[])
        {
        validate(13);
        System.out.println ("rest of the code...");
        }
}
```

Throws

- ✓ The Java throws keyword is used to declare an exception.
- ✓ If a method does not handle a checked exception, the method must declare it using the throws keyword.
- ✓ The throws keyword appears at the end of a method's signature.
- ✓ You can declare multiple exceptions.

Syntax:

```
return_type method_name () throws exception_class_name
{
//method code
}
```

Example:

```
import java.io.*;
class M
{
 void method()throws IOException
{
  throw new IOException("device error");
 }
}
public class Testthrows2
{
   public static void main(String args[])
{
    try
{
     M m=new M ();
     m.method();
    }
catch(Exception e)
{
System.out.println("exception handled");
}
    System.out.println("normal flow...");
  }
}
```

23. Explain Method Overriding with example.

Ans:

- When same method name, same argument and same return type is used in a sub class and superclass is called Method Overriding.
- Method Overriding can be done in different class.
- In a Method Overriding parameter must be same.
- Method overriding is the example of run time polymorphism.
- Return type must be same in Method overriding.
- For implementing method overloading we must require inheritance.

- Example:

```
Class Vehicle
{
 Void run()
{
System.out.println ("Vehicle is running");
}
}
Class Bike extends Vehicle
{
 Void run ()
{
System.out.println ("Bike is running safely");
}
 }
Class OverridingDemo
{
 public static void main(String args[])
{
 Bike obj = new Bike ();//creating object
 obj.run ();//calling method
 }
}
```

24. Define Package and write a step to create Package.

Ans:

Package:

- Packages are grouping of variety of classes and/or Interfaces together.
- Packages are conceptually similar as "class libraries "of other languages.
- Packages act as "containers" for classes.

Java packages are classified into two types

- Java API packages **or** system defined package
- User defined packages

Steps to create user-defined package

1. Declare the package at the beginning of file using ,

   Syntax : 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 all main source files are stored.
4. Store the listing file as the classname.java file in subdirectory you have created.
5. Compile that file. This will create .class file in subdirectory.

Example:

File1: saved in D:\javapro\package1\ClassA.java

```
package package1;
public class ClassA
{
public void displayA ()
{
                System.out.println ("Class A");
}
}
```

Save this class in file with name ClassA.java in package1

File2: saved in D:\javapro\PackageTest.java

```
import package1.classA;
class PackageTest
{
public static void main (String args [])
{
ClassA objA=new ClassA ();
objA. displayA ();
}
}
```

Output:

```
Class A
```

25. Explain Exception Handling.

Ans:

- An Exception is abnormal condition that is caused by a runtime error in the program.
- The purpose of exception handling is to provide a means to detect and report an "abnormal condition "so we can take proper actions.
- Error handling code consist of two segments

1) Detect errors and to throw exceptions

2) Catch the exceptions and take appropriate actions

- This mechanism performs following tasks in sequence.
  - Find the problem (Hit the Exception).
  - Inform that an error has occurred (Throw the Exception).
  - Receive the error information catch the exception)
  - Take corrective actions (Handle the Exception).
- **Exception Handling Code (Try Catch)**

  The basic concept of Exception handling are throwing an exception and catching it.

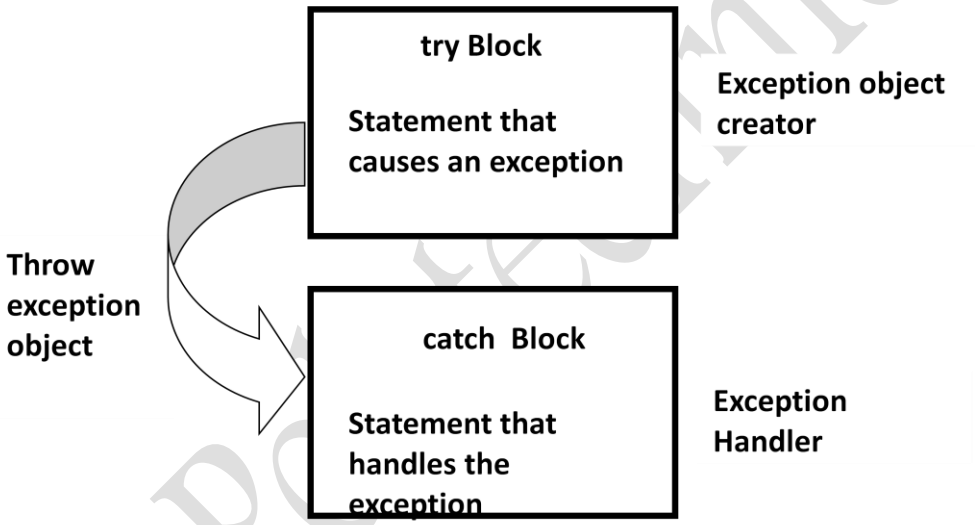


Fig. : Exception handling mechanism

26. Explain or Give the Difference between Types of Errors in Java.

Ans.

Compile time errors.	Run time errors.
Errors which are detected by javac at the compilation time of program are known as compile time errors.	Errors which are detected by java interpreter at the time of program are known as run time errors.
It is detected by javac (compiler) at compile time.	It is detected by java (interpreter) at rum time.
Whenever compiler displays an error, it will not create the .class file.	A program compiled successfully and creates a .class file but may not run properly.
It is also known as syntax error.	It is also called logical error.

Compile Time Errors are:	Run time errors are:
Missing semicolon Missing or mismatch of brackets in classes and methods Misspelling of identifiers and keywords Missing double quotes in strings Use of undeclared variables	Dividing an integer by zero. Accessing element that is out of the bounds of an array. Trying a store a value into an array of an incompatible class or type.

Example:
class Error1
{
public static void main(String args[])
{
System.out.println("Hello")
//Missing ;
}
}
	Example:
	class Error2
	{
	public static void main(String args[])
	{
	int a=10,b=5 ,c=5;
	int x = a / (b-c); // division by zero
	System.out.println("x=" + x);
	int y = a / (b+c);
	System.out.println("y=" + y);
	}}

27. Write a program for generating Exception "Divided by Zero".

Ans:

```
class Error3
{
 public static void main (String args[])
  {
  int  a=10,b=5,c=5;
  int x,y;
  try
  {
          X=a / (b-c);      //Exception here
  }
  Catch (ArithmeticException e)
  {
  System.out.println ("division by zero");
  }
  Y= a / (b +c );
```

```
    System.out.println ("y="+y);
}
}
```

Output:

```
Division by zero
y=1
```

28. Explain abstract keyword with example.

Ans:

- Abstract is a keyword in Java.
- In Java, abstract keyword can be used with class and method.
- When we want must define method overriding compulsory then the method is declared with the keyword abstract.
- CONDITION TO USE ABSTRACT KEYWORD
  - We cannot use abstract class for creating object directly.
  - The method of abstract class which is declared with abstract keyword must be redefined in its subclass.
  - We cannot declare abstract constructor or abstract static method.
- Example:

```
abstract class A
{
        abstract void callme();
        void callmeToo()
        {
                System.out.println("I AM NORMAL METHOD");
        }
}
class B extends A
{
void callme()
{
System.out.println("I AM ABSTRACT METHOD");
}
}
class Demo
{
```

```
        public static void main(String[] args)
        {
                B b1=new B();
                b1.callme();
                b1.callmeToo();
        }
}
```

29. Explain Dynamic method dispatch with example.

Ans:

- Dynamic dispatch is a mechanism by which a call to an overridden method is resolved at runtime rather than compile time.
- It is also known as runtime polymorphism.
- A super class reference variable can referred to a subclass object is known as up-casting.
- When an overridden method is called through a super class reference the determination of the method to be called is based on the object being referred by the reference variable.
- This determination is done at runtime.
- **Example:**

```
class A
{
    void callme()
    {
            System.out.println("METHOD OF CLASS A");
    }
   class B extends a
    {
    void callme()
    {
            System.out.println("METHOD OF CLASS B");
    }
}
class C extends A
{
    void callme()
```

```
    {
        System.out.println("METHOD OF CLASS C");
}
}
class DispatchDemo
{
    public static void main(String [] args)
    {
        A a1 = new A( );
        B b1= new B( );
        C c1= new C( );
        A r;
        r = a1;
        r.callme( );
        r=b1;
        r.callme( );
        r= c1;
        r.callme( );
    }
}
```