All

1. **What is JAVA ?**
   - ▼ **Ans**
     1. **Java is a programming language which is used to write some set of instruction to perform some specific tasks.**
     2. **Java is developed by Sun Microsystems for development of an applications.**
     3. **Java is a high level, platform independent, object oriented programming language which is use to develop the application like Standalone application, Web application, Third party application.**
     4. **Java is a Simple, Portable, Multithreaded, High Performance, Secured, Robust(Strong) language.**
     - ▼ **Standalone application**
       - **An application which is limited only to one single system is know as StandAlone application.**
       - **Ex: Any desktop application (Calculator, Notepad, Media player, MS paint, etc).**
     - ▼ **Web application**
       - **A Web application an application program that is stored on a remote server and delivered over the Internet through a browser interface.**
       - **Ex: Amezon.com**
     - ▼ **Third party application**
       -
     - ▼ **What is Software ?**
       - **Software is a set of instructions or programs used to operate computers and execute specific tasks.**

       Ex : For Software is "OS".
     - ▼ **What is an Application ?**
       - **Set of programs forms an application.**
         - ▼ **What is Program ?**
           - **Set of instruction (or) command forms a program.**
2. **What is Class and Object ?**
   - ▼ **Ans**
     - **Object :- Object is Real World Physical Entities which has its own States and Behaviors.**
       1. **State Represents the Properties of Object.**
       2. **Behavior Represents Functionality of an Object.**
     - **Class :- Class is a blue print design which describes the structure for the Object. It is also know as Logical Entity.**
2. **What are the components of Objects ?**
   - ▼ Ans

     Object has 2 components
     1. **State**
        - **State represents the properties of an Object.**
     2. **Behavior**
        - **Behavior represents action or functionality of an object.**

2. **What are the members of the Class ?**

▼ Ans

There 4 members are there in class

1. **States**
2. **Behaviors**
3. **Constructor**
4. **Blocks**

2. **What is Keywords ?**

▼ **Ans**

- **Keywords are collection of Pre-defined words and each keywords has specific purpose to serve in Java.**

2. **What is Identifier ?**

▼ **Ans**

- **Identifier is a name given to the components of a java program.**

2. **How to create an Object in Java ?**

▼ Ans

- **Using constructor declaring with new keyword and storing it in one variable with respective data type.**

2. **What is Declaration , Initialization , Direct Initialization and Object Reference ?**

▼ **Ans**

Declaration :-

- **The process of writing the properties is called as Declaration.**

Initialization :-

- **The process of assigning the value to the properties called as Initialization.**

Direct Initialization :-

- **The process of declaration and initialization of a properties at the same time is called Direct Initialization.**

Object Reference :-

- **The container which stores the hexadecimal value of the object address is called Object Reference.**

2. **What is Data-Types ?**

▼ **Ans**

Data-Type describes the what kind of data we are going to store it in memory allocation. 2types of Data-Type

1. **Primitive Data-Type**
    - **Primitive Data-Types are pre-defined data-types which are available in form of keywords.**
    - **Primitive Data-Types can store decimal , non-decimal , symbol, true and false. but they are not capable of storing hexadecimal value that is nothing but Object Address. hence we go to Non-primitive Data-types.**
2. **Non-Primitive Data-Type**
    - **Non-Primitive Data-Types can be created by creating classes and Interface.**
    - **class as two meaning blue print and primitive datatypes.**

2. **What is Parameter and Arguments ?**

▼ **Ans**

- **Parameter is container which can hold Arguments.**

- **Arguments are the value pass by the User.**

2. **How many ways can we initialize a Variable ?**

▼ **Ans**

5 Ways we can initialize a Variable

1. **Direct Initialization.**
2. **Object Reference.**
3. **Constructor.**
4. **Blocks.**
5. **Methods.**

2. **Explain System.out.println() ?**

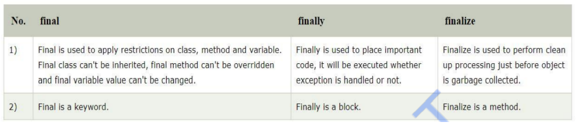
▼ **Ans**

- **System.out.println is a composed statement.**
- **print, println are the overloaded non-static methods of Printstream class which is present in the java.io package.**
- **out is a static object reference of System class which is present in java.lang package.**

3. Explain final, finally and finalize ?

▼ Ans

- **final** = final is a **keyword and is used as access modifier** in Java
- **finally** = Finally is a **block in Java used for Exception Handling**.
- **finalize** = Finalize is a **method in Java used for Garbage Collection.**



No.	final	finally	finalize
1)	Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.	Finally is used to place important code, it will be executed whether exception is handled or not.	Finalize is used to perform clean up processing just before object is garbage collected.
2)	Final is a keyword.	Finally is a block.	Finalize is a method.

4. What is difference between Continue and Break Statement ?

▼ Ans

- Break and Continue statements are the jump statements that are used to skip or terminate the loop immediately.

▼ Break Statement

- **Break statement stops the entire process of the loop.**
- Break also terminates the remaining iterations.

▼ Example Program

```
public static void main(String[] args)
  {
    for (int i = 0; i < 10; i++)
        {
      if (i == 5)
        break;
      System.out.println("Manu ->"+i);
    }
  }
----------------OUTPUT--------------
i: 0
i: 1
i: 2
i: 3
i: 4
```

▼ Continue Statement

- **Continue statement only stops the current iteration of the loop.**
- Continue doesn't terminate the next iterations; it resumes with the successive iterations.

▼ Example Program

```
public static void main(String[] args)
  {
    for (int i = 0; i < 5; i++)
    {
            if (i == 2)
                continue;
            System.out.println("Manu --> "+i);
    }
  }
-----------------OUTPUT-----------------------
Manu --> 0
Manu --> 1
Manu --> 3
Manu --> 4
```

5. How String is Immutable ?

▼ Ans

- The string is immutable means that String objects cannot be modified after they're created, but that String object reference can be changed. The String class has made final to avoid others to extend it and destroy its immutability.

▼ Advantage of String is Immutable ?

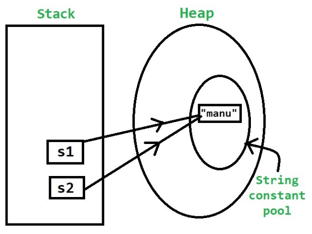
- It provides the security to the application. If the String is not immutable, any hacker can cause a security issue in the application by changing the reference of that String value. and also
- It saves the memory.

▼ Example Program (Creating a String by Literals)

- Another time if we create a same content of new reference of String by using Literals, it is not going to create new String Object. instead new reference is going to refer old same String Object. because old String content and new String is same. It is we can say as String is immutable.
- If we are creating a String in Literal way, String object is going to store in String constant pool.

▼ String constant pool

- String pool is nothing but a storage area inside heap memory where string literals stores. It is also known as String Intern Pool or String Constant Pool.



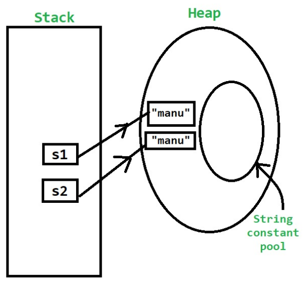
```
public static void main(String[] args)
  {
     String s1="manu";
     String s2="manu";

    System.out.println(s1==s2);
  }
----------OUTPUT----------
true
```

▼ Example Program (Creating a String by ‘new’ keyword)

- Whenever if we create String object of same content in ‘n’ number of by using ‘new’ keyword, it is going to create new String object irrespective of their content.
- If we are creating a String using new keyword, String object is going to store in heap memory.

  ▼ Heap memory

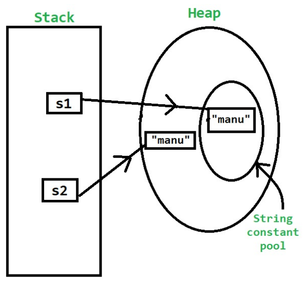
  - Heap memory used to store objects instantiated by applications running on the JVM.



```
public static void main(String[] args)
  {
     String s1=new String("manu");
     String s2=new String("manu");

    System.out.println(s1==s2);
  }
----------OUTPUT----------
false
```

▼ Example Program (Literals + new)

- Here s1 reference is going create their object in String constant pool and s2 reference is going to create their object in Heap area or memory. both object are different.



```
public static void main(String[] args)
  {
    String s1="manu";
    String s2=new String("manu");

    System.out.println(s1==s2);
  }
-----------------OUTPUT----------------
false
```

6. How can be a object dereferenced ?

▼ Ans

- when an object is created with one reference and that object is called by another reference also first reference will be dereferenced.
- when an object is created with one reference and that object is call with another object reference also first object reference will be dereferenced.

  ▼ Example Program

  ```
  public static void main(String[] args)
    {
        Runner3 r1 = new Runner3();
        Runner3 r2=r1;
    }
  ```

7. Difference between == and equals() method?

▼ Ans

- Both are used to compare two objects in Java

▼ ==

- It is an operator, it is used for “Reference Comparison” (address comparison) .
- If we are using Primitive values it will compare with content. But If we are using non-primitive it will Compare with reference.

  ▼ Example Program

  ```
  public static void main(String[] args)
    {
        int a1=5;
        int a2=5;
      System.out.println(a1==a2);//In primitive it compares the content

      Integer t1=new Integer(5);
      Integer t2=new Integer(5);

      System.out.println(t1==t2);//In non-primitive it compares the reference
    }
  ----------------------OUTPUT----------------------
  true
  false
  ```

▼ equals()

- It is a method. it is used for “content comparison”.
- This method should be used with Non-Primitive only.

  ▼ Example Program

  ```
  public static void main(String[] args)
    {
      String s1="manu";
      String s2="manu";
      System.out.println(s1==s2);//same object b'cause String is Immutable
      System.out.println(s1.equals(s2)); //content is same
      System.out.println();

      String s3="manu";
      String s4="manukm";
      System.out.println(s3==s4);//different object
      System.out.println(s3.equals(s4));//different object
      System.out.println();

      Integer t1=new Integer(5);
      Integer t2=new Integer(5);
      System.out.println(t1==t2);//different object
  ```

```
    System.out.println(t1.equals(t2));//content is same
  }
----------------------OUTPUT-----------------
true
true

false
false

false
true
```

▼ Example Given in interview

```
public class MainRunner2
{
  public static void main(String[] args)
  {
        String s1="java";
        String s2=s1;
        String s3=new String(s1);
        System.out.println(s1==s2);
        System.out.println(s1.equals(s2));
        System.out.println(s2==s3);
        System.out.println(s2.equals(s3));
  }
}
----------------OUTPUT------------------------
true
true
false
true
```

8. Can we make array as heterogenous ?

▼ Ans

- Yes, by using Generalization, it is possible

▼ Example Program

```
public static void main(String[] args)
  {
        Object manu[]={"manu km",1,1.1,true,'A'};
        System.out.println(Arrays.toString(manu));
  }
----------------OUTPUT-----------
[manu km, 1, 1.1, true, A]
```

9. Difference between For and For each loop?

▼ Ans

▼ For loop

- for loop is used as a general purpose loop.
- for-loop is present from JDK1.
- In for loop we can iterate in both decrement or increment order.
- we can print array elements as original order and reverse order.

▼ Example Program

```
public static void main(String[] args)
  {
        Object manu[]={"manu km",1,1.1,true,'A'};

        for(int i=0; i < manu.length; i++)
        {
          System.out.print(manu[i]+" ");
        }

        System.out.println();

        for(int i=manu.length-1; i >=0; i--)
        {
          System.out.print(manu[i]+" ");
        }
  }
--------------------OUTPUT----------------
manu km 1 1.1 true A
A true 1.1 1 manu km
```

▼ For-each loop / Enhanced for -loop / Advanced for loop

- foreach loop is used for arrays and collections.
- for-each loop is present from JDK5.
- In for-each loop we can iterate only in increment order.
- we can print array elements as original order but not as reverse order.

▼ Example Program

```
public static void main(String[] args)
  {
        Object manu[]={"manu km",1,1.1,true,'A'};

        for ( Object O : manu )
        {
          System.out.print(O+" ");
        }
  }
```

13. What is WORA ?

▼ Ans

- WORA (Write Once Run Anywhere). This means a programmer can develop java code on one system and can expect to run on any other platform without any adjustments. this is all possible because of JVM.

14. What is Scanner class ?

▼ Ans

- It will take input from the user.
- Scanner is a pre-defined class in java, it is present in the java.util package. this util package has many class in that one is a Scanner .
- Inside the Scanner class many non-static methods are there:-

1. nextBoolean()
2. nextByte()
3. nextShort()
4. nextInt()
5. nextLong()
6. nextFloat()
7. nextDouble()
8. next()
9. nextLine()

33. How to prevent or avoid variable shadowing ?

▼ Ans

- By using “this” keyword.

26. Why we use final keyword ?

▼ Ans

- We can use final keyword in 3 ways :-

▼ final Variables

- final variable is variable that we can assign a value only one time.
- If we don’t want to change variable value then we should assign final to that Variable. A final variable cannot be modified.

▼ Real Time Example

- PI ———present——> Math class ———present——> lang package

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println(Math.PI);
  }
}
----------------OUTPUT------------
3.141592653589793
```

▼ Example Program

```
public class Test
{
  public static void main(String[] args)
  {
    final int manu = 10 ;
    manu=15;

    System.out.println(manu);
  }
}
---------OUTPUT--------
CTE
```

▼ final Method

- final method cannot be overridden.
- If you don’t want to change any method implementation to child class then assign final to the parent method.

▼ Real Time Example

▼ Example Program

```
public class Test
{
  final public void display()
  {
    System.out.println("Punneth raj Kumar");
  }
}
class Demo extends Test
{
  public void display()
  {
    System.out.println("Ragnar");
  }
  public static void main(String[] args)
  {
    Demo d = new Demo();
    d.display();

  }
}
----------OUTPUT---------
CTE
```

▼ final Class

- final class cannot be inherited.
- If we don't want to give permission to child-class to extend parent-class, then assign final keyword to that parent class.

▼ Real Time Example

- String class ———Present——> lang package

▼ Example Program

```
final public class Test
{

}
class Demo extends Test
{

}
----------OUTPUT-----
CTE
```

27. What is Actual parameter and Formal parameter ?

▼ Ans

▼ Actual Parameter

- Actual Parameter are the value or variables in the parenthesis of method call.

▼ Formal Parameter

- Formal Parameter are the variables in the parenthesis of method signature. it receives the values from the method call.

▼ Example

```
public class Test
{
  static int display(int x,int y)//Formal parameter
  {
    return x+y;
  }
  public static void main(String[] args)
  {
     int manu=display(10,20);//Actual parameter
     System.out.println(manu);
  }
}
--------------OUTPUT--------
30
```

32. What is Data Mis-handling ?

▼ Ans

- Data Mis-handling means using any information in a way it's not supposed to be used.

36. Difference between Method and Constructor ?

- ▼ Ans
  - ▼ Method
    - Method is a member of a class and also known as behavior of an object. Method is a set of instruction which will be used to perform some specific task.
      - ▼ Advantage / Why we use Method ?
        - We can achieve Code Reusability.
        - We can avoid Code Duplication.
        - Modification becomes simple.
  - ▼ Constructor
    - Constructor is a member of a class and also know as special member. Constructor is mainly used for 2 purpose that is object creation and Initialization of properties. the major advantage of constructor is object creation and initialization properties can be done at the same time.
  - ▼ Difference

Method	Constructor
* We need call method explicitly.	* Constructor is automatically called when an object is created.
* Method name can be anything.	* Constructor name same as class name.
* We can declare return type for a method.	* There is no return type for constructor.
* There is no method provided by Compiler.	* There is always default constructor provided by Compiler.
* A Method can be inherited by subclasses	* A Constructor cannot be inherited by subclasses

37. Where to find java source code ?
    - ▼ Ans
      - **The JDK source code is inside the src.**zip
38. What is the output of S.o.p(null); ?
    - ▼ Ans
      - Compile Time Error
39. What is Open Source Software and Closed Software ?
    - ▼ Ans
      - ▼ Open Source Software (OSS)
        - ▼ Definition of OSS
          - A software whose source code is freely distributed with a license to study, change and further distributed to anyone for any purpose is called open source software.
          - Dedicated programmers improve upon the source code and share the changes within the community.
        - ▼ OSS provides these advantages to the users due to its thriving communities:
          1. Security
          2. Affordability
          3. Transparent
          4. Interoperable on multiple platforms
          5. Flexible due to customizations

6. Localization is possible

- ▼ Explanation of OSS
  - If we wants to develop a software, but we cannot afford to do it from start or do not have the time or do not have the expertise or we do not have that much of money to invest so we can take open source software that is related to our requirement and then we can just develop or customize it according to our own needs then we can use further.
- ▼ Types of OSS
  - ▼ Freeware
    - Freeware is available free of cost for use and distribution but it cannot be modified because source code is not available. It is still called open source software because we do not have to pay for it.

      Ex: Google Chrome, Skype
  - ▼ Shareware
    - Shareware is initially free and can be distributed but it needs to be paid for after a stipulated period of time because source code is not available.

      Ex: Win RAR

▼ Proprietary Source Software (PSS) or Closed Software

- ▼ Definition of PSS
  - It is a software that be used only by obtaining a license from its developer after paying fees for it is called Proprietary Source Software.
  - Individual or company can own such proprietary software
  - Source code is closely guarded secret

    Ex:- Microsoft Windows
- ▼ Major Restriction like:-
  - No further distribution
  - Number of users that can use it
  - Type of computer it can be installed on, example multitasking or single user, etc.

39. What is difference between Open Source and Free Software ?

▼ Ans

- "free software" and "open source software" refer to essentially the same set of licenses, but they arrive at that set different routes. (The results aren't perfectly identical, but the differences are unlikely to matter broadly.)

Open Source	**Free Software**
* Open source is a development methodology.	* Free software is social movement.
* Every open-source software is not free software.	* Every free software is open source
* Open-source software occasionally imposes some restrictions on users	* No restrictions are imposed on free software
Ex: internet browsers Mozilla Firefox, chrome and Apache HTTP Server	Ex: MySQL relational database and Apache web server

40. What is Varargs (Variable Arguments)?

▼ Ans

▼ Definition

- Varargs is a short name for variable arguments.
- Variable Arguments (Varargs) is a method that takes a multiple variable number of arguments. It is introduce in JDK 1.5 version.

▼ Adanvantage of Varargs Method

- It is not required to declare multiple method, we can declare only one method so that length of the code going to be reduced and reusability is achieved.

▼ How to declare Varargs methd

- Declaration —> m1(int… x)
- calling —> m1( ) , m1(10) , m1(10,20) , m1(2,8,3,6,) (we can pass values any number time)

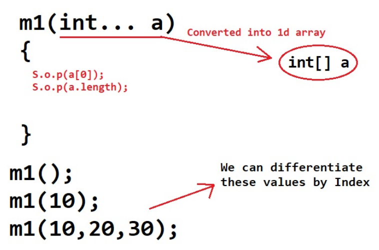
▼ Example Programs

▼ Program 1

```
public class Test
{
  public static void display(int... x)
  {
    System.out.println("manu km");
  }
  public static void main(String[] args)
  {
    display();
    display(10);
    display(10,20);
    display(10,20,30);
    display(10,20,30,40);
  }
}
--------------OUTPUT---------------
manu km
manu km
manu km
manu km
manu km
```

▼ Program 2

- Internally Var-arg parameter will be converted into one dimensional array.
- Hence within the var-arg method we can differentiate values y using Index.



```
public class Test
{
  public static void m1(int... a)
  {
    System.out.println(a[0]);
    System.out.println(a[1]);
    System.out.println(a[3]);
    System.out.println(a[2]);
    System.out.println(a.length);
  }
  public static void main(String[] args)
  {
    m1(10,20,30,40);
  }
}
-----------------OUTPUT----------------
10
20
40
30
4
```

Program 3

```
public class Test
{
  public static void display(int... a)
  {
    int sum = 0;
    for( int O : a)
    {
      sum = sum + O;
    }
    System.out.println(sum);
  }
  public static void main(String[] args)
  {
    display(10,20,30,40);
  }
}
--------------------OUTPUT---------------------------
100
```

Loopholes of var-arg methods

Case 1

1. Display(int . . . x) —> (Valid)
2. Display(int . . .x) —> (Valid)
3. Display(int. . .x) —> (Valid)
4. Display(int x . . .) —> (Invalid)
5. Display(int .   . . x) —> (Invalid)
6. Display(int . x . .) —> (Invalid)

Case 2

- We can mix Var-arg parameter with normal parameter also.

  Ex: Display(int x , int . . . y) , Display(int x , String . . . y)

Example Program

```
public class Test
{
  public static void display(int x , int... a)
  {
    System.out.println(x);
    System.out.println(a[0]);
    System.out.println(a[1]);
    System.out.println(a.length);
  }
  public static void main(String[] args)
  {
    display(10,20,30,40);
  }
}
-----------------OUTPUT--------------------
10
20
30
3
```

▼ Case 3

- If we mix normal parameter with Var-arg parameter then var-arg parameter should be last parameter. Otherwise we will get compilation error.

  Ex: Display(int . . . y , int x) , Display(String . . . y , int x)

  ▼ Example Program

```
public class Test
{
  public static void display(int... a , int x)
  {
    System.out.println(x);
    System.out.println(a[0]);
    System.out.println(a[1]);
    System.out.println(a.length);
  }
  public static void main(String[] args)
  {
    display(10,20,30,40);
  }
}
---------------------OUTPUT-------------------------
CTE
```

▼ Case 4

- Within Var-arg method we can take only one var-arg parameter. i.e., we can't declare more than one var-arg parameters inside var-arg method. if we do that we will get a compilation error.

  Ex: Display(int . . . y , int . . . x) , Display(String . . . y , int . . . x)

▼ Case 5

- In general var-arg method will get least priority i.e., if no other method matched then only var-arg method will get the chance. It is exactly same as default case inside switch.

  ▼ Example Program

```
public class Test
{
  public static void display(int... x)
  {
    System.out.println("Var-arg Method ");
  }
  public static void display(int x)
  {
    System.out.println("Normal Method ");
  }
  public static void main(String[] args)
  {
    display(10);
  }
}
------------------OUTPUT------------------------
Normal Method
```

▼ Case 6

- With in a class if we are declaring var-arg parameter method then in the same class we can't declare corresponding one-dimensional array method. If we do Compile Time Error.

  ▼ Example Program

```
public class Test
{
  public static void display(int[] x)
  {
    System.out.println("Var-arg Method ");
  }
  public static void display(int... x)
  {
    System.out.println("Normal Method ");
  }
  public static void main(String[] args)
  {
    display(10);
  }
}
```

- ▼ Equivalence between 1d array and Var-arg parameter
  - ▼ Case 1
    - Where ever one-dimensional array present we can replace with var-arg parameter

      Ex: display(int[] x) ——replace——> display(int . . . x) ⇒ (Valid)
  - ▼ Case 2
    - Where ever Var-arg method parameter present we can't replace with one-dimensional array.

      Ex: display(int . . . x) ——replace——> display(int[] x) ⇒ (Invalid)

42. How do you delete an object in java ?
    - ▼ Ans
      - You should remove the references to it by assigning null or leaving the block where it was declared. After that, it will be automatically deleted by the garbage collector (not immediately, but eventually).
    - ▼ Example Program

```
Object a = new Object();a = null;
 // after this, if there is no reference to the object.
// it will be deleted by the garbage collector
```

44. How do you convert String to Integer? How do you convert Integer to String ?
    - ▼ Ans
    - ▼ String —> int
      - ▼ Without Using Inbuilt Method
        - This code is works both Negative and Positive also

```
static int convert(String str)
  {
    int num=0,i=0;
    boolean neg=false;
    if(str.charAt(0) == '-')
    {
      neg = true;
      i = 1;
    }
    for(i=i; i<str.length(); i++)
    {
    num = num * 10 + (str.charAt(i)-48);
    }
    if(neg)
    {
      num = -num;
    }
    return num;
  }
    public static void main(String... args)
    {
      String str = "10225";
      int x=convert(str);
      System.out.println(x+1);
    }
------------OUTPUT-------------
10226
```

▼ Using Inbuilt Method

We have 4ways to convert String into int

▼ Integer.parseInt()

- Integer.parseInt() —>parseInt() is static method—→present—>Interger class—> lang package.

▼ Example Program

```
public static void main(String... args)
    {
      String str = "5";
        String str2 = "10";

       int x = Integer.parseInt(str);
       int y = Integer.parseInt(str2);

       System.out.println(x+y);
    }
---------OUTPUT-------------------
15
```

▼ Integer.valueOf()

- Integer.valueOf() —>valueOf() is static method—→present—>Interger class—> Lang package.

▼ Example Program

```
public static void main(String... args)
    {
      String str = "5";
        String str2 = "10";

       int x = Integer.valueOf(str);
       int y = Integer.valueOf(str2);

       System.out.println(x+y);
    }
---------OUTPUT-------------------
15
```

▼ intValue()

▼ DecimalFormat

▼ int —> String

▼ Without Using Inbuilt Method

- This code is works both Negative and Positive

```
public static void main(String... args)
    {
      int x = 1000;
      String str=x+"";
      System.out.println(str);
    }
------------OUTPUT-------------
"1000"
```

▼ Using Inbuilt Method

We have 3ways to Convert int to String

- ▼ Integer.toString()
  - Integer.toString() —> toString() is static method —> Integer class —> lang package.
    - ▼ Example Program

```
public static void main(String... args)
    {
      int x = 5;

      String str=Integer.toString(x);

      System.out.println(str+1);
    }
-------------OUTPUT-----------
"51"
```

- ▼ String.valueOf()
  - String.valueOf() —> valueOf() is static method —> String class —> lang package.
    - ▼ Example Program

```
public static void main(String... args)
    {
      int x = 5;

      String str=String.valueOf(x);

      System.out.println(str+1);
    }
-------------OUTPUT---------------
"51"
```

- ▼ String.format()
  - String.format(,) —> format(,) is static method —> String class —> lang packege.
    - ▼ Example

```
public static void main(String... args)
    {
      int x = 5;

      String str=String.format("%d", x);

      System.out.println(str+1);
    }
```

47. Explain the different types of object creation in java ?

- ▼ Ans
  - There are 6 ways to create an Object in Java
    - ▼ Using 'new' keyword

```
public class Test
{
  public Test()
  {
    System.out.println("1st way Object is Created");
  }
  public static void main(String... args)
  {
    Test O1 = new Test();
  }
}
```

```
----------OUTPUT-----------
1st way Object is Created
```

▼ Using reflection

```
public class Test
{
  public Test()
  {
    System.out.println("2st way Object is Created");
  }
  public static void main(String... args) throws Throwable, Throwable, Throwable
  {
    Test O2 = (Test) Class.forName("org.Practise.Work_SpaceApp.Test").newInstance();
                                      ////We Should Pass FullyQualiedName
  }
}
----------------------OUTPUT-----------
2st way Object is Created
```

▼ Using clone() method

- clone() method is present in Object class and it is used to create a copy of an existing object. in order to the clone() method corresponding class should have implements a Cloneable Interface which is again a Marker Interface.

```
class Test implements Cloneable
{
  public Test()
  {
    System.out.println("2th Object is Created");
  }
  public static void main(String[] args) throws Exception
  {
    Test o1 = new Test();
    Test o2 = (Test) o1.clone();
  }
}
-------------------------------OUTPUT-------------------------------
3th Object is Created
```

▼ Using ClassLoader

```
public class Test
{
  public Test()
  {
    System.out.println("4th Object is Created");
  }
  public static void main(String[] args) throws Exception, IllegalAccessException, ClassNotFoundException
  {
    Test o4 = (Test) Test.class.getClassLoader().loadClass("org.Practise.Work_SpaceApp.Test").newInstance();
  }
}
-------------------------OUTPUT------------------------------------
4th Object is Created"
```

▼ Using Contructor class from java.lang.reflect

```
import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
public class Test
{
  public Test()
  {
    System.out.println("5th Object is Created");
  }
  public static void main(String[] args) throws Throwable, IllegalAccessException, IllegalArgumentException, InvocationTar
  {
    Class<Test> class1 = Test.class;
    Constructor<?> constructor = class1.getDeclaredConstructors()[0];
    Test o5 = (Test) constructor.newInstance();
  }
}
-------------------------------OUTPUT-----------------------------------------------------------------
5th Object is Created
```

- Using Deserialization

49. How to make variable as constant ? like PI=3.142 ?
    - Ans
        - By making that variable as a final static. (Combination of final and static keyword will make Variable as Globally Constant data member).
            - Note:-
                - If the container or data holder can able change its value then that type of container we call it as a Variable.
                - If the container or data holder constant or cannot able change its value then that type of container we call it as a Data-Member.
50. Can we Overload final methods? Give an Example for final overloaded method ?
    - Ans
        - Yes, We can Overload the final methods,
        - Real Time Example
            - wait() methods of Object class.
                1. wait() with zero parameter
                2. wait(long, int)
                3. wait(long)
48. What is Relationship ? Difference between Has-A relationship and IS-A relationship ?
    - Ans
        - Relationship
            - The connection (Association) between two object is know as the relationship.
                - Types of Relationship
                    - HAS-A Relationship
                        - If one object is dependent on another object then it is known a Has-A relationship

                        (or)

                        - Whenever an object of one class is used in another class, it is called HAS-A relationship.
                            - Types of Has-A Relationship
                                - We can achieve HAS-A Relationship in 2 ways:
                                    - Aggregation
                                        - The dependency between two object such that one object can exist without the other is know as Aggregation.

                                          Ex: Cab-ola, Train-online ticket booking
                                    - Composition

- The dependency between two object such that one object cannot exist without the other is know as Composition.

  Ex: Car-Engine, Human-Oxygen

▼ IS-A Relationship

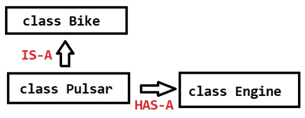
- Whenever one class inherits another class, it is called an IS-A relationship.
- In an IS-A relationship, the child object will acquire all properties of the parent object, and the child object will have its own extra properties.
- IS-A relationship is achieved with the help of Inheritance.

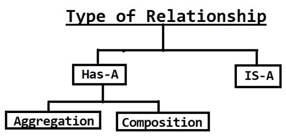
▼ Types of IS-A Relationship

- We can achieve IS-A relationship in 2 ways:

▼ Generalization

▼ Specialization





43. Difference between String, StringBuilder, StringBuffer ?

▼ Ans

▼ <u>String class</u>

- <u>String is a set of character</u> in java. String <u>objects are immutable</u> which means a constant and cannot be changed once created.
- String class has made as final, present in the lang package and implements Serializable, Comparable and CharSequence.
- We can create String object in 2 ways

▼ Using Literals

- If we create String object using Literal way, that String object is going to store in String constant pool.

```
class Test
{
  public static void main(String[] arg)
  {
      String str = "manu"; //Literal way of String Object

      System.out.printl(str);
  }
}
-----------------OUTPUT------------------------------
manu
```

▼ Using 'new' keyword or constructor

- If we create String object using constructor, that String object is going to store in Heap memory.

```
class Test
{
  public static void main(String[] arg)
  {
      String str = new String("manu"); //By Using constructor

      System.out.printl(str);
  }
}
-----------------OUTPUT-----------------------------
manu
```

- **StringBuffer class**
  - StringBuffer in Java represents a mutable sequence of characters.
  - StringBuffer class has made as final, present in the lang package and extends AbstractStringBuilder implements Serializable and CharSequence.
- **StringBuilder class**
  - StringBuilder in Java represents a mutable sequence of characters.
  - StringBuilder class has made as final, present in the lang package and extends AbstractStringBuilder implements Serializable and CharSequence.
- Difference between String, StringBuilder, StringBuffer

String	StringBuffer	StringBuilder
* String object is Immutable	* StringBuilder object is mutable.	* StringBuffer object is mutable.
* If the content is fixed and won't change frequently then we should go for String.	* If the content is not fixed and keep on changing but Thread Safety is required then we should go for StringBuffer.	* If the content is not fixed and keep on changing and thread safety is not required then we should go for StringBuilder.
	* Thread safe	* Not Thread safe.
	* Single-threaded because every methods in StringBuilder are Synchronized.	* Multi-threaded because every methods in StringBuilder is non-Synchronized.
	* performance wise is low	* performance wise is fast
	* Introduced in JDK 1.0 version (Legacy)	* Introduced in JDK 1.5

41. Can you Explain cloning ?
    - Ans
      - The process of creating an exact duplicate object is called cloning
        - Purpose of cloning
          - The main purpose of coning is to maintain back-up copy and preserve state of an object at a particular instance.
        - How can we perform cloning
          - By using clone() method of Object class, we can perform cloning.
            - clone()

              protected native java.lang.Object clone() throws java.lang.CloneNotSupportedException;
          - We can perform cloning only for cloneable objects. An Object is said to be cloneable if only if the corresponding class implements Cloneable interface. Cloneable Interface present in lang package and it does not contains any method it is a marker interface.
        - Example Program

```
class Test implements Cloneable //maker Interface //Runtime Exception
{
  int i=8;
  int j=12;
  public static void main(String[] args) throws CloneNotSupportedException
  {
    Test t1=new Test();
    Test t2=(Test) t1.clone(); //Checked Exception
    t2.i=88;
    t2.j=1000;

    System.out.println(t1.i+"   "+t1.j);
    System.out.println(t2.i+"   "+t2.j);
  }
}
------------------------------OUTPUT---------------------------
8   12
88   1000
```

▼ Shallow cloning and deep cloning

- Object class clone() method is Shallow cloning. If we want deep cloning we should override clone() method.
- Refer durgha softs videos —> (https://youtu.be/L-p-RGTNrZg)

46. Write a program to override equals(), hascode() and toString() method ?

▼ Ans

▼ toString()

- We use toString() method to get String representation of an object.

▼ Purpose of overriding toString()

- Whenever we are trying print any object reference, internally toString() method will be called then toString() method will return String representation of Hexadecimal value.

  return format : classname@Hexadecimal

- Based on our requirement we can override toString() method to provide our own String representation. it return a String based on state of an Object.

▼ Note:

- In all Wrapper classes, Collection Classes, String class, StringBuffer class, StringBuilder class toString() method is already Overriden

▼ Example Program (Without Overridng toString())

```
package org.Practise.Work_SpaceApp;
public class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }
  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    System.out.println(t1);
  }
}
--------------------------OUTPUT-------------------------------
org.Practise.Work_SpaceApp.Test@15db9742
```

▼ Example Program (Overridng toString())

```
class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }
  public String toString()
  {
    return this.name+" "+this.id;
  }
  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    System.out.println(t1);
  }
}
-----------------OUTPUT-------------------
manu 125
```

▼ Important Example

```
package org.Practise.Work_SpaceApp;
import java.util.*;
class Test
{
  String name;
  int id;
  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }

  public static void main(String[] args)
  {
    Test t1 = new Test("manu",125);
    System.out.println(t1);

    String s1 = new String("manu");
       System.out.println(s1);

    Integer I1 = new Integer(10);
       System.out.println(I1);

    ArrayList<String> A = new ArrayList<String>();
       A.add("manu");
       A.add("km");
       System.out.println(A);
  }
}
---------------------OUTPUT-------------------------------------
org.Practise.Work_SpaceApp.Test@15db9742
manu
10
[manu, km]
/*( In all Wrapper classes, Collection Classes, String class,
     StringBuffer class, StringBuilder class toString() method
          is already Overriden )*/
```

▼ equals(Object)

- We use equals() method to check equality of two object.
- The return type of equals(Object) method is boolean.
- for equals() method we should pass any Object Reference.
- equals(Object) method is used to compare the reference of the two Objects.

▼ Purpose of equals(Object)

- We should override to equals(Object) method to compare the state or properties (or) contents of two Object instead of comparing reference of two Object.
  - ▼ Note:
    - If equals(Object) method is not overridden, it compares the reference of two objects similar to '==' operator.
    - If equals(Object) method is overridden, it compares the state of an Objects in such case comparing the reference of the two object is possible only by '==' operator.
      - ▼ Design Tip
        - In equals method compare the state of a current object with the passed object with the passed object by down-casting the passed object.
  - ▼ Example Program (If we not Override)

```
class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }

  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    Test t2=new Test("manu",125);
    System.out.println(t1==t2);
    System.out.println(t1.equals(t2));
  }
}
---------------OUTPUT-----------------------
false
false
```

  - ▼ Example Program (If we Override)

```
class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }
    public boolean equals(Object O)
    {
      Test t=(Test) O;
      return this.name.equals(t.name) && this.id==t.id;
    }
  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    Test t2=new Test("manu",125);
    System.out.println(t1==t2);
    System.out.println(t1.equals(t2));
  }
}
------------------OUTPUT--------------------------
false
true
```

  - ▼ Example Program (If we not Override)—>Durga soft

```
class Test
{
  String name;
  int rollno;
  Test(String name,int rollno)
  {
    this.name=name;
    this.rollno=rollno;
  }
  public static void main(String[] args)
  {
    Test t1=new Test ("manu",101);
    Test t2=new Test ("km",102);
    Test t3=new Test ("manu",101);
    Test t4 = t1 ;
    System.out.println(t1.equals(t2));
    System.out.println(t1.equals(t3));
    System.out.println(t1.equals(t4));
  }
}
-------------------OUTPUT----------------------------
false
false
true
```

▼ Example Program (If we Override) —>Durga soft

```
import java.util.*;
class Test
{
  String name;
  int rollno;
  Test(String name,int rollno)
  {
    this.name=name;
    this.rollno=rollno;
  }
  public boolean equals(Object O)
  {
    String name1 = this.name;
    int rollno1 = this.rollno;

    Test s = (Test) O;
    String name2 = s.name;
    int rollno2 = s.rollno;

    if(name1.equals(name2) && rollno1==rollno2)
    {
      return true;
    }else {
      return false;
    }
  }
  public static void main(String[] args)
  {
    Test t1=new Test ("manu",101);
    Test t2=new Test ("km",102);
    Test t3=new Test ("manu",101);
    Test t4 = t1 ;
    System.out.println(t1.equals(t2));
    System.out.println(t1.equals(t3));
    System.out.println(t1.equals(t4));
  }
}
----------------------OUTPUT---------------------------
false
true
true
---------------------Simplification of equals()--------------
public boolean equals(Object O)
  {
    Test s = (Test) O;
    if(name.equals(s.name) && rollno==s.rollno)
    {
      return true;
    }else {
      return false;
    }
  }
-----------:Durga soft note:----------------------------------
* To make above equals more efficient we have to write
   the following code at begining inside equals() method if(O==this)
   r{ eturn true;}. According to the if both the references pointing
   to same object then without performing any comapriso .equals() return
   true directly.
public boolean equals(Object O)
  {
    if(O==this)
    {
        return true;
    }
    Test s = (Test) O;
    if(name.equals(s.name) && rollno==s.rollno)
    {
      return true;
    }else {
      return false;
    }
  }
```

▼ Important Example 1

```
import java.util.*;
class Test
{
  String name;
  int id;
  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }

  public static void main(String[] args)
  {
    Test t1 = new Test("manu",125);
    Test t2 = new Test("manu",125);
    System.out.println(t1.equals(t2));

    String s1 = new String("manu");
    String s2 = new String("manu");
       System.out.println(s1.equals(s2));

    Integer I1 = new Integer(10);
    Integer I2 = new Integer(10);
       System.out.println(I1.equals(I2));

    ArrayList<String> A = new ArrayList<String>();
       A.add("manu");
       A.add("km");
    ArrayList<String> B = new ArrayList<String>();
       A.add("manu");
       A.add("km");
       System.out.println(A.equals(B));
  }
}
---------------------OUTPUT------------------------------------
false
true    ///In String class .equals() method is already overriden
true    // for content comparison that is why it is true. but in
        /// StringBuffur.equals() method is not overriden. we
        /// should explicitly override.
```

▼ Important Example 2

```
class Test
{
  public static void main(String[] args)
  {
    String s1 = new String("manu");
    String s2 = new String("manu");
    System.out.println(s1.equals(s2));

    StringBuffer s3 = new StringBuffer("manu");
    StringBuffer s4 = new StringBuffer("manu");
    System.out.println(s3.equals(s4));
  }
}
-------------------OUTPUT-------------------------
true
false
/// StringBuffur.equals() method is not overriden. we
        /// should explicitly override for comaprision
```

▼ hashCode()

- For every object an Unique number is generated by JVM is called hashCode. hashcode will not represent address of an object. JVM will use hashcode while saveing object into hashing related data structures like HashTable, HashMap, HashSet, etc
- Advantages of saving object based on hashCode is Search operation will become easy. (The most powerful search Algorithm up to today is Hashing)
- The return type of hashcode() method is int.
- hashCode() method will give the hashCode(or)unique number based on the reference of an object.

▼ Purpose of Overriding hashcode() method

- We should override hashcode() method to return a unique integer based on the state of two objects instead based on reference of two Objects.
- If equals(Object) method is overridden, then it is necessary to override the hashCode() method.

▼ Design Tip

- hashCode() method should return an number based on the state of Object.

▼ Example Program (without Overriding hashCode())

```
class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }

  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    Test t2=new Test("manu",125);
    System.out.println(t1.hashCode());
    System.out.println(t2.hashCode());
  }
}
-------------------------------OUTPUT------------------
366712642
1829164700
```

▼ Example Program (Overriding hashCode())

```
class Test
{
  String name;
  int id;

  public Test(String name,int id)
  {
    this.name=name;
    this.id=id;
  }
  public int hashCode()
  {
    return id+name.hashCode();
  }
  public static void main(String[] args)
  {
    Test t1=new Test("manu",125);
    Test t2=new Test("manu",125);
    System.out.println(t1.hashCode());
    System.out.println(t2.hashCode());
  }
}
----------------------OUTPUT-------------
3344088
3344088
```

▼ Override these 3 methods and write one Program

```

```

Method name	Access Modifier	Return type	Arguments	Description
toString	public	String	No	String representation of an Object
equals(Object)	public	boolean	Object	Content Comparison
hashCode()	public	int	no	Unique id/no for identify an Object.

45. What is the equality contract in java ?

▼ Ans

Contract between .equals() method and hashCode() method:

- If 2 objects are equal by .equals() method compulsory their hashcodes must be equal (or) same. That is If r1.equals(r2) is true then r1.hascode()==r2.hashcode( ) must be true.
- If 2 objects are not equal by .equals() method then there are no restrictions on hashCode() methods. They may be same (or) may be different. That is If r1.equals(r2) is false then r1.hashCode()==r2.hashCode() may be same (or) may be different.
- If hashcodes of 2 objects are equal we can't conclude anything about .equals() method it may returns true (or) false. That is If r1.hashCode()==r2.hashCode() is true then r1.equals(r2) method may returns true (or) false.
- If hashcodes of 2 objects are not equal then these objects are always not equal by .equals() method also. That is If r1.hashCode()==r2.hashCode() is false then r1.equals(r2) is always false.

  ▼ Note:

  - To maintain the above contract between .equals() and hashCode() methods whenever. we are overriding .equals() method compulsory we should override hashCode() method. Violation leads to no compile time error and runtime error but it is not good programming practice.

34. What is Operator ?

▼ Ans

- Operator in java is a symbol that is used to perform operations.

  ▼ Types of Operators Precedence (According to their priority arranged)

  - There are 9 types of Operator in Java : -

    ▼ Unary Operator

Unary Operator are —> ([ ], x++, x- -, ++x, - -x, ~, !)

▼ Arithmetic Operator

Arithmetic Operator are —> (*, /, %, +, -)

▼ Shift Operator

Shift Operator —> (>>, >>>, <<)

▼ Comparison Operator

Comparison Operator —> (<, <=, >, >=, instanceof)

▼ Equality Operator

Equality Operator —> (==, !=)

▼ Bitwise Operator

Bitwise Operator —> (&, ^, |)

▼ Short Circuit Operator

Short Circuit Operator —> (&&, | |)

▼ Conditional Operator

Conditional Operator —> (?:)

▼ Assignment Operator

Assignment Operator —> (=, +=, -=, *=, , ,etc)

▼ Note:

- In java we have only operator precedence but not operand precedence before applying any operator all operands will be evaluated from left to right.