**TASKS – 03/01/2018**

1. **Can abstract class have constructors in Java?**

An abstract class can have a constructor but we cannot create an object of abstract class. So when we inherit that abstract class in subclass we can pass values to its constructor through super() method in subclass. So using super we can pass values in constructor of abstract class

1. **Can abstract class implements interface in Java? do they require to implement all methods?**

Yes abstract class can implement interface. When we implement an interface to an abstract class, its means that the abstract class inherited all the methods of the interface. As, it is not important to implement all the method in abstract class however it comes to abstract class, so the abstract class can left some of the method in interface without implementation here. But, when this abstract class will inherited by some concrete class, they must have to implements all those unimplemented method there in abstract class.

**3)**  **Can abstract class be final in Java?**

No. Reason is for final classes we can't create child class whereas for abstract classes we should create child class to provide implementation.

**4)  Can abstract class have static methods in Java?**

|  |  |
| --- | --- |
|  | In Java we can have a static method in an abstract class. This is allowed because that method can be called directly, even if we do not have an instance of the abstract class.  Abstract class A{  Static void demo(){}  } |
|  | A.demo(); |

**5)  Can you create instance of abstract class?**

No, you cannot create an instance of an abstract class because it does not have a complete implementation.

**6)  Is it necessary for abstract class to have abstract method?**

We can have an abstract class without Abstract Methods as both are independent concepts. Declaring a class abstract means that it can not be instantiated on its own and can only be sub classed. Declaring a method abstractmeans that Method will be defined in the subclass.

**7)  Difference between abstract class and interface in Java?**

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can have abstract and non-abstract methods. | Interface can have only abstract methods. Since Java 8, it can have default and static methods also. |
| 2) Abstract class doesn't support multiple inheritance. | Interface supports multiple inheritance. |
| 3) Abstract class can have final, non-final, static and non-static variables. | Interface has only static and final variables. |
| 4) Abstract class can provide the implementation of interface. | Interface can't provide the implementation of abstract class. |
| 5) The abstract keyword is used to declare abstract class. | The interface keyword is used to declare interface. |
| 6) Example: public abstract class Shape{ public abstract void draw(); } | Example: public interface Drawable{ void draw(); } |

**8)  When do you favor abstract class over interface?**

In below situations we can use abstract class over interface.

1. We want to share code among several closely related classes.
2. We expect that classes that extend abstract class have many common methods or fields or require access modifiers other than public (such as protected and private).

**9)   What is abstract method in Java?**

An *abstract method* is a method that is declared without an implementation as below.

abstract void abs(int a, int b);

If a class includes abstract methods, then the class itself *must* be declared abstract, as in:

public abstract class A {

abstract void B();

}

**10) Can abstract class contains main method in Java?**

Yes abstract class can have main method as below.

Class Test

{

  int x = 10;

  public void display()

  {

    System.out.println("Hello");

  }

}

public abstract class Demo

{

public static void main(String args[])

{

   Test t1 = new Test();

   System.out.println("From abstract class main(): " + t1.x);

   t1.display();

}

}

**11) What is static block in java?**

Static block is used for initializing the static variables. This block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program which means the first static block executes before second static block. That’s the reason, values initialized by first block are overwritten by second block.

**12) What is the need of static block?**

Static block can be used where a class has static variables that require complex initialization.

**13) Can we overload static methods in java?**

Yes. We can have two or more static methods with same name, but differences in parameters. For example, consider the following Java program.

|  |
| --- |
| public class Test {      public static void A() {          System.out.println("Test.A() called ");      }      public static void A(int a) {          System.out.println("Test.A(int) called ");      }      public static void main(String args[])      {          Test.A();          Test.A(10);      }  } |

**14) Can we call super class static methods from sub class?**

**15) What is the difference between final and static keywords?**

Static Keyword  
  
->Static keyword is used to do better memory management.   
->We can use static keyword with variable, method, class.   
->If you declare method using static keyword then you don't need to create an object to call that method.   
->If you declare variable as a static then only once memory is created for that variable in class area and not every time when object is created

Final Keyword:   
  
->Final Keyword is used to restrict the user. It means:   
->If you declare variable as static then you cannot change its value.   
->If you declare method as static then you cannot override that method.   
->If you make any class as final, you cannot extend it.

**16) Write a note on covariant return type with example code.**

Before JDK version 5, it was not possible to [override](https://www.geeksforgeeks.org/overriding-in-java/) a method by changing the return type. When we override a parent class method, the name, argument types and return type of the overriding method in child class has to be exactly same as that of parent class method. Java 5.0 onwards it is possible to have different return type for a overriding method in child class, but child’s return type should be sub-type of parent’s return type. Overriding method becomes variant with respect to return type.

An example shown below.

package pack1;

class A {}

class B extends A {}

class Base

{

A fun()

{

System.*out*.println("Base fun()");

return new A();

}

}

class Derived extends Base

{

B fun()

{

System.*out*.println("Derived fun()");

return new B();

}

}

public class Covariant

{

public static void main(String args[])

{

Base base = new Base();

base.fun();

Derived derived = new Derived();

derived.fun();

}

}

**17) Write a note on Enum with example code.**

**Enum in java is a** data type that contains fixed set of constants.

It can be used for days of the week (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY and SATURDAY. The java enum constants are static and final implicitly. It is available from JDK 1.5.

Java Enums are classes that have fixed set of constants.

* enum improves type safety
* enum can be easily used in switch
* enum can be traversed
* enum can have fields, constructors and methods
* enum may implement many interfaces but cannot extend any class because it internally extends Enum class

class EnumExample1{

public enum Season { WINTER, SPRING, SUMMER, FALL }

public static void main(String[] args) {

for (Season s : Season.values())

System.out.println(s);

}}

**18) Write a note on use of super keyword and super() method.**

The super keyword in java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Use of java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

**19) Write a code to implement abstraction using interface.**

interface abs

{

public void method1();

public void method2();

}

class Demo implements abs

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[])

{

abs obj = new Demo();

obj.method1();

}

}

**20) Write a Java program to sort a numeric array and a string array.**

import java.util.Arrays;

public class Sort {

public static void main(String[] args){

int[] a = {

34,54,344,32,54,21};

String[] s = {

"Java",

"Python",

"PHP",

"C#",

"C Programming",

"C++"

};

System.out.println("Original numeric array : "+Arrays.toString(a));

Arrays.sort(a);

System.out.println("Sorted numeric array : "+Arrays.toString(a));

System.out.println("Original string array : "+Arrays.toString(s));

Arrays.sort(s);

System.out.println("Sorted string array : "+Arrays.toString(s));

}

}

**21)Write a Java program to sum values of an array.**

class SumOfArray{

public static void main(String args[]){

int[] a = {10, 20, 30, 40, 50, 10};

int sum = 0;

for( int i : a) {

sum = sum+i;

}

System.out.println("Sum of array elements is:"+sum);

}

}

**22)Write a Java program to remove a specific element from an array.**

import java.util.Arrays;

public class Remove {

public static void main(String[] args) {

int[] a = {25, 14, 56, 15, 36};

System.out.println("Original Array : "+Arrays.toString(a);

int removeIndex = 1;

for(int i = removeIndex; i < my\_array.length -1; i++){

a[i] = a[i + 1];

}

System.out.println("After removing the second element: "+Arrays.toString(a));

}

}

**23)Write a Java program to reverse an array of integer values.**

public class ReverseArray {

    public static void main(String[] args) {

        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        System.out.println("Array before reverse:");

        for (int i = 0; i < numbers.length; i++) {

            System.out.print(numbers[i] + " ");

        }

        for (int i = 0, j = numbers.length - 1; i < j; i++, j--) {

            int temp = numbers[i];

            numbers[i] = numbers[j];

            numbers[j] = temp;

        }

        System.out.println("\nArray after reverse:");

        for (int i = 0; i < numbers.length; i++) {

            System.out.print(numbers[i] + " ");

        }

    }

}

24) Write a Java program to find the duplicate values of an array of integer values.

import java.util.Arrays;

public class Duplicate {

public static void main(String[] args)

{

int[] a = {1, 2, 5, 5, 6, 6, 7,8,8,2};

for (int i = 0; i < a.length-1; i++)

{

for (int j = i+1; j <a.length; j++)

{

if ((a[i] == a[j]) && (i != j))

{

System.out.println("Duplicate Element : "+a[j]);

}

}

}

}

}