1)  Can abstract class have constructors in Java?

A: Yes, it can have a constructor and the constructor is defined and behaves just like any other class's constructor. But, as the abstract classes cannot be directly instantiated, only extended, so the use of constructor is therefore always from a subclass's constructor.

2) Can abstract class implements interface in Java? do they require to implement all methods?

A: In Java, an abstract class can implement an interface, and not provide implementations of all of the interface's methods. It is the responsibility of the first concrete class that has that abstract class as an ancestor to implement all of the methods in the interface.

3)  Can abstract class be final in Java?  
A: No a class cannot be marked "abstract" as well as "final". Marking a class "abstract" means it contains partial implementation of methods and hence depends on its subclasses to provide concrete implementation. Marking the class "final" means that no other class can extend it and hence abstract classes cannot be marked final.

4)  Can abstract class have static methods in Java?

A: In Java you can have a static method in an abstract class. But I cannot have an abstract method which is static.

5) Can you create instance of abstract class?

A: No, we cannot create an instance of an abstract class.

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

A: No, it is not necessary for abstract class to have abstract method. We can have an abstract class without any abstract methods.

7) Difference between abstract class and interface in Java?

|  |  |
| --- | --- |
| Abstract class | Interface |
| 1) Abstract class can have abstract and non-abstract methods. | 1) 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. | 2) Interface supports multiple inheritance. |
| 3) Abstract class can provide the implementation of interface. | 3) Interface can't provide the implementation of abstract class. |
| 4)  The abstract keyword is used to declare abstract class. | 4) The interface keyword is used to declare interface. |

8) When do you favor abstract class over interface?

A: Consider using abstract classes if any of these statements apply to your situation:

1. You want to share code among several closely related classes.
2. You expect that classes that extend your abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).
3. You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.

Consider using interfaces if any of these statements apply to your situation:

1. You expect that unrelated classes would implement your interface. For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.
2. You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.
3. You want to take advantage of multiple inheritances.

9) What is abstract method in Java?

A: A method which is declared without an implementation is called an abstract method in java.

For example: abstract void add(int a, int b);

10)  Can abstract class contains main method in Java ?

A: Yes, abstract class can contain main method in Java, as main method is static and it does not require an instance to invoke it.

11) what is static block in java?

A: Static block is a block of code which is executed when the class gets loaded by a classloader. It is meant to do initialization of static members of the class.

12)  What is the need of static block?

A: Constructor is invoked while creating an instance of the class. Static block is invoked when a classloader loads this class definition, so that we can initialize static members of this class. We should not be initializing static members from constructor as they are part of class definition not object. Hence, we require static block to initialize static members of the class.

13)  Can we overload static methods in java?

A: Yes, we can overload static methods in java.

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

A: Yes, we can call super class static methods from sub class.

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

A: Once the final variable has been assigned a value, it cannot be changed. Final methods cannot be overridden. And final classes cannot be extended.

Static means it belongs to the class not an instance, this means that there is only one copy of that variable/method shared between all instances of a particular Class.

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

A: The covariant return type specifies that the return type may vary in the same direction as the subclass. 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.

class A{

A get(){return this;}

}

class B extends A{

B get(){return this;}

void message(){System.out.println("welcome to covariant return type");}

public static void main(String args[]){

new B().get().message();

}

}

As we can see in the above example, the return type of the get() method of class A is A but the return type of the get() method of class B is B. Both methods have different return type but it is method overriding. This is known as covariant return type.

17) Write a note on Enum with example code.

A: Enum in java is a data type that contains fixed set of constants. The java enum constants are static and final implicitly. Points to remember for Java Enum:

* 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 EnumEx{

public enum Season { WINTER, SPRING, SUMMER}

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.

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

The super keyword is used for the following:

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

19)  Write a code to implement abstraction using interface.

interface MyInterface{

void display(String name);

}

public class AbstractionUsingInterface implements MyInterface{

@Override

public void display(String name) {

System.out.println("Hello "+name);

}

public static void main(String args[]) {

MyInterface mi = new AbstractionUsingInterface();

mi.display("Pranil");

}

}

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class SortArray {

public static void main(String args[]) throws IOException

{

int numarr[], n;

String stringarr[];

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter number of elements in the numeric array");

n = Integer.parseInt(br.readLine());

numarr = new int[n];

System.out.println("Enter number of elements in the string array");

n = Integer.parseInt(br.readLine());

stringarr = new String[n];

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

{

System.out.println("Enter numeric element "+(i+1));

numarr[i] = Integer.parseInt(br.readLine());

}

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

{

System.out.println("Enter string element "+(i+1));

stringarr[i] = br.readLine().toLowerCase();

}

System.out.println("Original numeric array before sorting:"+Arrays.toString(numarr));

Arrays.sort(numarr);

System.out.println("Numeric array after sorting:"+Arrays.toString(numarr));

System.out.println("Original string array before sorting:"+Arrays.toString(stringarr));

Arrays.sort(stringarr);

System.out.println("String array after sorting:"+Arrays.toString(stringarr));

}

}

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class SumOfArray {

public static void main(String args[]) throws IOException

{

int arr[], sum = 0;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter number of elements in the array");

int n = Integer.parseInt(br.readLine());

arr = new int[n];

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

{

System.out.println("Enter element "+(i+1));

arr[i] = Integer.parseInt(br.readLine());

}

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

{

sum += arr[i];

}

System.out.println("Sum of array = "+sum);

}

}

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class RemoveElement {

public static void main(String args[]) throws IOException {

int arr[], n, arrelement, index = 0;

boolean flag = false;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter number of elements in the array");

n = Integer.parseInt(br.readLine());

arr = new int[n];

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

{

System.out.println("Enter element "+(i+1));

arr[i] = Integer.parseInt(br.readLine());

}

System.out.println("Enter the element to be removed from the array");

arrelement = Integer.parseInt(br.readLine());

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

{

if(arr[i] == arrelement)

{

index = i;

flag = true;

break;

}

}

for(int i = index; i < arr.length - 1;i++)

{

arr[i] = arr[i+1];

}

if(flag == true)

{

System.out.println("Element deleted...");

System.out.println("Array after removing "+arrelement+" :");

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

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

}

}

else

System.out.println("Entered element not found");

}

}

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class ReverseAnArray {

public static void main(String args[]) throws IOException {

int arr[], revarr[];

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter number of elements in the array");

int n = Integer.parseInt(br.readLine());

arr = new int[n];

revarr = new int[n];

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

System.out.println("Enter element " + (i + 1));

arr[i] = Integer.parseInt(br.readLine());

}

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

revarr[j]= arr[i];

}

System.out.println("Original array:");

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

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

}

System.out.println("");

System.out.println("Reversed array:");

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

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

}

}

}

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class ArrayDuplicates {

public static void main(String args[]) throws IOException {

int arr[];

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter number of elements in the array");

int n = Integer.parseInt(br.readLine());

arr = new int[n];

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

System.out.println("Enter element " + (i + 1));

arr[i] = Integer.parseInt(br.readLine());

}

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

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

if(arr[i] == arr[j])

System.out.println("Duplicate element: "+arr[i]+" found");

}

}

}

}