**Java interview questions and answers**

1. Advantages of Java.

* Java is Object Oriented programming language
* Java is easy to learn
* Java is open source
* Java is platform independent
* Java is robust and secure

1. What is JVM, JRE and JDK

|  |  |  |
| --- | --- | --- |
| **JVM** | **JRE** | **JDK** |
| Java Virtual Machine | Java Runtime Environment | Java Development Kit |
| Java is platform independent. | JRE is platform dependent. | JDK is platform dependent. |
| JVM is subset of JRE | It is subset of JDK | Superset of JRE |
| It provides java byte code and provides an environment for executing. | It is S/W bundle which provides java class libraries with components to run java code. | It is S/W development kit to develop applications in java. |
| It bundled with both JRE and JDK. | It contains only environment to execute source code. | It runs with Installer. |

1. **Oops concepts with programs.**
   1. **Object**

Ans. An object is an instance of a class. Objects are allocated memory space whenever they are created.

* 1. **Class**

Ans. Group of objects is called Class. It’s a blueprint of the object. When class is created no memory is allocated.

* 1. **Inheritance**

Ans. It is possible to inherit attributes and methods from one class to another class.

Inheritance concept in two categories:

* **Sub class** (child): the class that inherits from another class.
* **Superclass** (parent): the class that is inherited from.

Syntax: class child extends parent {

//program

}

Types of Inheritance:

1. Single Inheritance :

It inherits only one sub class is derived from one super class .

**Program:**

class Textbook{

void Telugubook(){

System.out.println("language book");

}

class Material extends Textbook{

void English(){

System.out.println("language book");

}

}

public class School{

public static void main(String args[]){

Material m = new Material();

m.Telugubook();

m.English();

}

}

}

1. **Multi-level Inheritance**

It inherits the data from super class to subclass again subclass to subclass.

**Example:**

class Shape {

public void display() {

System.out.println("Inside display");

}

}

class Rectangle extends Shape {

public void area() {

System.out.println("Inside area");

}

}

class Cube extends Rectangle {

public void volume() {

System.out.println("Inside volume");

}

}

class Tester {

public static void main(String[] arguments) {

Cube cube = new Cube();

cube.display();

cube.area();

cube.volume();

}

}

1. **Hierarchical Inheritance**

where more than one class is derived from the parent class,there is a single base class and multiple derived classes.

**Example:**

class Parent {

void parentMethod() {

System.out.println("This is a method from the Parent class.");

}

}

class Child1 extends Parent {

void child1Method() {

System.out.println("This is a method from Child1 class.");

}

}

class Child2 extends Parent {

void child2Method() {

System.out.println("This is a method from Child2 class.");

}

}

public class Main {

public static void main(String[] args) {

Parent parentObj = new Parent();

Child1 child1Obj = new Child1();

Child2 child2Obj = new Child2();

parentObj.parentMethod();

child1Obj.parentMethod();

child1Obj.child1Method();

child2Obj.parentMethod();

child2Obj.child2Method();

}

}

1. Hybrid Inheritance
   1. **Polymorphism**

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

**Example:**

class Animal {

public void animalSound() {

System.out.println("The animal makes a sound");

}

}

class Pig extends Animal {

public void animalSound() {

System.out.println("The pig says: wee wee");

}

}

class Dog extends Animal {

public void animalSound() {

System.out.println("The dog says: bow wow");

}

}

* 1. **Abstraction**

Abstraction is the process of hiding the internal data ans shows only essential data to the user.

interface Animals{

void one();

void two();

}

public class Lion implements Animal{

public void one(){

System.out.println("king of the forest");

}

public void two(){

System.out.println("animals");

}

}

public class Forest{

public static void main(String args[]){

Animal ani = new lion();

ani.one();

ani.two();

}

}

* 1. **Encapsulation**

Wrapping up of a data into a single unit is called Encapsulation.

1. Difference between Interface and abstraction.

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| Abstract class doesn’t support Inheritance. | Interface support multiple inheritance. |
| Abstract class doesn’t support multiple inheritance. | Interface supports Multiple inheritance. |

1. **Operators**

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Category** | **Precedence** |
| Unary | postfix | expr++ expr-- |
|  | prefix | ++expr –expr +expr -expr |
| Arithmetic | multiplicative | \*/% |
|  | addictive | +- |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >=instance of |
|  | equality | == != |
| Bitwise | bitwise AND | & |
|  | bitwise exclusive OR | ^ |
|  | bitwise inclusive OR | | |
| Logical | logical AND | && |
|  | logical OR | || |
| Ternary | ternary | ?: |
| Assignment | assignment | = += -= \*= %= &= ^= |= <<= >>= >>>= |

1. Conditions and Loops(If, If else, for, while)

Loops:

**For loop:**

We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

1. **Initialization**: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.
2. **Condition**: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return Boolean value either true or false. It is an optional condition.
3. **Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.
4. **Statement**: The statement of the loop is executed each time until the second condition is false.

**Syntax**:

For (initialization; condition; increment/decrement) {

//statement

}

**Example:**

Public class ForExample {

public static void main (String args []) {

for (int i=0; i<5; i++) {

Systemm.out.println (i);

}

}

}

**Output:**

1

2

3

4

5

**WhileLoop:**

The [while loop](https://www.scaler.com/topics/while-loop-in-java/) is used when the number of iterations is not known but the terminating condition is known. Loop is executed until the given condition becomes false. While loop is also an **entry-controlled** loop as the condition is checked before entering the loop.

**Example:**

Public class WhileFactorial{

Public static void main(String args[]){

Int fact=1,tyemp=0,num=5;

Temp=num;

While(temp!=0){

Fact=fact\*num

}

}

}

**do-while Loop:**

do-while loop is used to looping a part of program repeatedly. Until the condition gets true.

Syntax:

Do{

//code to execute

}

While{

}(condition)

**Example:**

public class DoWhileloop{

public static void main(String args[]){

int i=1;

do{

System.out.println(i);

i++;

}while(i<=10);

}

}

1. Exception handling with programs

Ans. Exception Handling means handling runtime errors such as IOexception,Class Not Found Exception etc.,

Types of java Exceptions are:

1. **Checked exception**

Checked exceptions are checked at compile time. The classes that directly inherit the throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException.

1. **Unchecked exception**

Unchecked exceptions are not checked at compile time but checked at runtime. The classes that inherit the RuntimeException are known as unchecked exceptions. For example Arithmetic exception ,ArrayIndexOutOfBoundsException etc.

1. **Error**

Error is irrecoverable some example of errors are OutOfmemoryError,VirtualMachineError etc.

**Java Exception Keywords:**

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| try | “try” keyword is used to specify a block of code.we can’t use try keyword alone. It must follows catch or finally. |
| catch | “catch” keyword is used to handle exception.we can’t use alone it follows finally. |
| finally | Final block is used to execute necessary code. |
| throw | “throw” keyword is used to throw an exception. |
| throws |  |

**Example:**

Public class EcxeptionExample{

Public static void main (String args[]){

Try{

Int data=100/0;

}

Catch(ArthematicException e){

System.out.println(e);

}

System.out.println(“rest of the code is…”)

}

}

1. Static, Final and Finally keywords
2. String operations with programs
   1. **Substring**

Part of string is called Substring.

**Example:**

public class SubString{

public static void main(String args[]){

String s=”Virat Kohli”;

System.out.println(“Original string is :”+s);

System.out.println(“after String starting is :”+s.substring(5));//Kohli

System.out.println(“string is :”+s.substring(0,5));//Virat

}

}

* 1. **Repeated char in a String**

We remove the repeated elements from a string

public class DuplicateString{

public static void main(String args[]){

String string1=”Duplicate Element”;

Int count;

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

Count=1;

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

count++;

string[j]=’0’;

}

}

If(count>1 && string[i] != ‘0’){

System.out.println(string[i]);

}

}

}

* 1. **Reverse string**

Public class Reversetring{

Public static void main(String args[]){

String s1=”ReverseString”;

String s2=””;

For(int i=s1.length()-1;i>0;i--){

s2+=s1.charAt(i);

}

System.out.println(“input string is :”+s1);

System.out.println(“ReverseS String is :”+s2);

}

}

* 1. **Compare string**

Comparing two strings is same or not.

**Example:**

public class Comparision{

public static void main(String args[]){

String s1=new String(”Virat”);

Stringf s2= new String(“Dhoni”);

System.out.println(“Comparing of S1 and S2 is :”+s1.equals(s2));

}

}

1. **Arrays with operations**
   1. Slice
   2. Push
   3. Merge two arrays

Merging of two arrays into a single array.

**Example**:

class Merging{

public static void main(String args[]){

int a[]={2,3,4};

int a1=a.length;

int b[]={5,6,7};

int b1=b.length;

int c1=a1+b1;

int c[]=new int [c1];

for(int i=0;i<a1;i++){

c[i]=a[i];

}

for(int i=0;i<b1;i++){

c[a1+i]=b[i];

}

for(int i=0;i<c1;i++){

System.out.print(c[i]);

}

}

}

* 1. **Remove duplicate with for loop**
  2. **Sort** **array**

**Example**:

Public class SortingArray{

Public static void main(String args[]){

Int arr[]=new int[]{22,4,11,65,98};

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

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

Int temp=0;

If(arr[i]>arr[j]){

Temp=arr[i];

Arr[i]=arr[j];

Arr[j]=temp;

}

}

System.out.println(arr[i]);

}

}

}

1. **Collections**
   1. **List**

**Example**:

Public class ListArray{

Public static void main(String args[]){

List<String>players= Arrays.asList{“virat”,”dhoni”,”sachin”,”rohit”};

For(int i=0;i<players.length();i++){

System.out.println(players.get(i));

}

}

}

* 1. **Set**

**HashSet:**

Hashset contains only unique elements .it allow null values.it doesnot maintain insertion order.

Example:

Import java.util.\*;

Public class HashSet{

Public static void mani(String args[]){

HashSet<string> set=new HashSet<String>();

Set.add(“virat”);

Set.add(“Dhoni”);

Set.add(“rohit”);

Set.add(“rahul”);

System.out.println(“the elements are:”+set);

Set.remove(“rohit”);

System.out.println(“after removing elements are :”+set);

HashSet<String>set1= new hashSet<String>();

Set1.add(“jadeja”);

Set1.add(“iyer”);

Set.addAll(set1);

System.out.printrln (“updated list is :”+set);

Set.removeif (str->str.contains (“rohit”));

System.out.println (“after removing if method ():”set);

Set. Clear ();

System.out.println (“after clear () elements are: “set);

}

}

**TreeSet:**

TreeSet contains only unique elements.

It maintains assending order.

It doesn’t allow null values.

**Example:**

Public class TreeSet{

Public static void main(String args[]){

TreeSet<String>tree= new TreeSet<>();

Tree.add(“virat”);

Tree.add(“Dhoni”);

Tree.add(“rohit”);

Tree.add(“rahul”);

System.out.println(“the elements are :”+tree);

}

}

* 1. Map

**Hash Map:**

Hash map contains values based on the key.

It doesn’t maintain order.

It contains only unique values.

**Example:**

Public class HashMap{

Public static void main(String args[]){

HashMap<Integer,String>hash= new HashMap<Integer,String>();

Hash.put(1,”virat”);

Hash.put(2,”Dhoni”);

Hash.put(3,”rohit”);

System.out.println(“hashmap elements are :”+hash);

For(Map.Entry m=map.entrySet()){

System.out.println(m.getKey()+””+m.getValue());

}

}

}

**Output:**

Hashmap elements are:

1 virat

2 Dhoni

3 rohit

**TreeMap:**

Tree map contains values based on the key.

It maintain’s ascending order.

It contains only unique values.

It doesn’t having null key but having many null values.

**Example:**

import java.util.\*;

public class Treemap{

public static void main(String args[]){

TreeMap<Integer,String> tree = new TreeMap<Integer,String>();

Tree.put(2,”Dhoni”);

Tree.put(3,”rohit”);

Tree.map(1,”virat”);

Tree.map(4,”rahul”);

For(Map.Entry m= map.entrySet()){

System.out.println(m.getKey()+””+m.getValue());

}

}

}

**Output:**

1 virat

2 Dhoni

3 rohit

4 rahul

* 1. How to create object and usage with example

Public class CreateObject{

Void show();{

System.out.println(“Creating objeect”);

}

Public static void main (String args[]){

CreatObject obj= new CreateObject();

Obj.show();

}

}

1. Threads
   1. **What is thread:**

A thread in Java is the direction or path that is taken while a program is being executed. Generally, all the programs have at least one thread, known as the main thread,that is provided by the JVM or Java Virtual Machine at the starting of the program's execution.

* 1. **Lifecycle of threads**

**New:**

A newly created thread that has not yet started the execution.

**Runnable:**

Either running or ready for execution but it’s waiting for resource allocation

**Running:**

A running thread is a thread that has been started and is currently executing its instructions

**Dead:**

A thread dies naturally when its run () method exits normally.

**Blocked:**

A thread moves to the blocked state when it wants to access an object that another thread has locked. Once that resource is available (unlocked) for the thread, it is no longer blocked and moves to the runnable state.

* 1. **Advantages of Threads**
* We can execute multiple applications of an application at a time.
* Reduce the complexity of big applications.
* Helps to improve the performance of an application.
* Utilizes the max resources of multiprocessor systems.
* Better user interface in case of GUI based applications.