**Tek School**

**Java Interview Questions**

**Part 2**

1. **Difference between Serialization and Deserialization in Java.**

**Ans:** **These are the difference between serialization and deserialization in java:**

| **Serialization** | **Deserialization** |
| --- | --- |
| Serialization is the process which is used to convert the objects into byte stream | Deserialization is the opposite process of serialization where we can get the objects back from the byte stream. |
| An object is serialized by writing it an ObjectOutputStream. | An object is deserialized by reading it from an ObjectInputStream. |

<https://www.quora.com/What-is-the-difference-between-serialization-and-deserialization-in-core-Java>

1. **What is SerialVersionUID?**

**Ans:** Whenever an object is Serialized, the object is stamped with a version ID number for the object class. This ID is called the SerialVersionUID. This is used during deserialization to verify that the sender and receiver that are compatible with the Serialization.

<https://www.geeksforgeeks.org/serialversionuid-in-java/>

1. **Write a program to swap 2 numbers without a temporary variable?**

int a=10;

int b=20;

a=a+b;// first this should be there a=10+20=30

b=a-b; // b= 30-20=10

a=a-b; //a=30-10=20

System.out.println(a);

System.out.println(b);

**//swap strings**

String x="Hello";

String y="TekSchool";

x=x+y; //HelloTekSchool

y=x.substring(0,(x).length()-y.length());

x=x.substring(y.length());

System.out.println(x);

System.out.println(y);

<https://www.geeksforgeeks.org/swap-two-numbers-without-using-temporary-variable/>

1. **Find out how many alpha characters present in a string?**

String given="wefeqf878979797fewfewrf879797efds&^&^\*^\*^";

String replaced=given.replaceAll("[a-zA-Z]", "");

int alphaChar=given.length()-replaced.length();

System.out.println(alphaChar);

<https://stackoverflow.com/questions/7252142/how-to-check-how-many-letters-are-in-a-string-in-java/7252166>

1. **How to find whether given number is odd number?**

int a=22;

if (a%2!=0) {

System.out.println(a+" is an odd number");

}else {

System.out.println(a+" is not an odd number");

}

<https://stackoverflow.com/questions/7342237/check-whether-number-is-even-or-odd/7342253>

1. **How to find out the part of the string from a string? What is substring? Find number of words in string?**

String a="You Made it to Tek School Interview Session";

System.out.println(a.substring(35));

String [] words=a.split(" ");

System.out.println(words.length);

for (String string : words) {

System.out.println(string);

}

<https://www.geeksforgeeks.org/searching-for-character-and-substring-in-a-string/>

1. **Write java program to reverse String? Reverse integer?**

**Reverse String:**

String a= "Hello Amerca";

// 1 method

StringBuffer sb=new StringBuffer(a);

System.out.println(sb.reverse());

//2 method

String reverse="";

for (int i=a.length()-1; i>=0; i--) {

reverse=reverse+a.charAt(i);

}

System.out.println(reverse);

// 3 method

String reverse1="";

for (int i=a.length(); i>=1; i--) {

reverse1=reverse1+a.substring(i-1, i);

}

System.out.println(reverse1);

**Reverse Integer**

int a=123;

// 1 method - cheapest way

String numbers=String.valueOf(a);

StringBuffer sb=new StringBuffer(numbers);

System.out.println(sb.reverse());

Look for other solution if interested

int currentNum=12345;

int reversedNum=0;

while(currentNum!=0) {

reversedNum=reversedNum\*10+currentNum%10;

currentNum=currentNum/10;

}

System.out.println(reversedNum);

<https://www.programmingsimplified.com/java/source-code/java-program-reverse-number>

1. **Write a program to sort array in ascending order?**

//1 method

int [] a= {12,24,2,4,1};

Arrays.sort(a);

System.out.println(Arrays.toString(a));

<https://www.sanfoundry.com/java-program-sort-array-ascending-order/>

1. **How can you convert a String in to an array? Array to String?**

Converting String to an Array

String a="TekSchoolAmerica";

char[] array=a.toCharArray();

System.out.println(array.length);

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

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

}

Converting Array to String

System.out.println(Arrays.toString(array));

<https://www.quora.com/How-do-I-convert-string-array-to-string-in-Java>

1. **Verify whether given two strings are equal?**

String a="Hello";

String b= "Hello";

1 method

if (a.equals(b)) {

System.out.println("String "+a+" and String "+b+" are equal");

}else {

System.out.println("String "+a+" and String "+b+" are not equal");

}

2 method

if (a.compareTo(b)==0) {

System.out.println("String "+a+" and String "+b+" are equal");

}else {

System.out.println("String "+a+" and String "+b+" are not equal");

}

<https://www.geeksforgeeks.org/compare-two-strings-in-java/>

1. **What is the difference between String and StringBuffer? What is mutable and immutable?**

The most important difference between String and StringBuffer in java is that String object is

immutable whereas StringBuffer object is mutable.

The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

By immutable, we mean that the value stored in the String object cannot be changed.

For example we cannot reverse string directly, only through using StringBuffer class.

<https://stackoverflow.com/questions/2971315/string-stringbuffer-and-stringbuilder>

**immutability vs. mutability**

String is immutability class it means once we are creating String objects it is not possible to

perform modifications on existing object. (String object is fixed object)

StringBuffer is a mutability class it means once we are creating StringBuffer objects on that

existing object it is possible to perform modification.

class Test {

public static void main(String[] args) {

String a="Hello";

String b="Hello";

StringBuffer sb=new StringBuffer("Hello TekSchool");

a=a.concat(" TekSchool");

System.out.println(a);

sb=sb.append(" America");

System.out.println(sb);

}

}

<https://www.interviewcake.com/concept/java/mutable>

1. **What are the collections you have used?**

Mostly in my current project we use ArrayList (There are other collections Set and Maps - just for your awareness and you can mention that you don't use them in your project).

If you know it please explain to the interviewer.

<http://www.java67.com/2013/01/difference-between-set-list-and-map-in-java.html>

1. **What are arrays and list? Difference between them?**

1. Arrays are fixed in size but ArrayLists are dynamic in size.

2. Arrays can store homogeneous elements whereas collections can store both. Example: in

Array we can store either int or String or boolean whereas in Array list we can store all of

them together

3. To find the size on an Array we use ArrayName.length and for arrayList we use

ArrayListName.size()

**ArrayList:**

ArrayList al=new ArrayList();

al.add("JackMa");

al.add("JackMa");

al.add(10);

for (int i=0; i<al.size(); i++) {

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

}

**Array:**

[] array=new String[3];

array[0]="John";

array[1]="Jordon";

array[2]="Jordon";

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

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

}

<https://www.quora.com/What-is-the-difference-between-an-array-and-an-array-list>

1. **What is constructor? Use of constructor in class?**

Constructor gets invoked when a new object is created.

Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler

builds a default constructor for that class.

Constructors must have the same name as the class and can not return a value.

They are only called once while regular methods could be called many times.

public BaseProject() {

try {

prop = new Properties();

FileInputStream fis = new FileInputStream("filePath.properties");

prop.load(fis);

} catch (IOException e) {

e.getMessage();

}

}

OR

public LoginPage() {

//initialize element using driver and this means current class objects

PageFactory.initElements(driver, this);

}

<https://www.javatpoint.com/java-constructor>

1. **How can we access variable without creating an object instance of it? What is Instance**

**variables and how you use it?**

By declaring variable as a static we can access it from different class - those variables called class

variables and also known as static variables are declared with the **static** keyword in a class, but

outside a method, constructor or a block. Whereas, **Instance** variables are declared in a class, but

outside a method, constructor or any block.

//class variables

public class TestBase {

public static WebDriver driver;

public static Properties prop;

//instance variable

public class LoginStepDefinitions {

WebDriver driver;

<https://www.dummies.com/programming/java/what-is-the-static-keyword-in-java/>

1. **How to print without main method "hello"?**

You can say that personally you don’t find a practical use of such way, but you know it can be done

using static block.

static

{

System.out.println("hello");

}

NOTE: Static block has the highest priority in java.

So, any thing that is written in static block is executed first.

<https://stackoverflow.com/questions/8605137/printing-message-on-console-without-using-main-method>

1. **How can you handle exceptions? How many catch blocks can we have?**

By using **“Try Catch Block”.**

There can be any number of catch block for a single try block and It is not necessary that each try

block must be followed by a catch block. It should be followed by either a catch block or a finally

block.

However only the catch block encountered first on the call stack that satisfies the condition for the

exception will be executed for that particular exception, rest will be ignored.

public TestBase() {

try {

prop=new Properties();

FileInputStream fis=new FileInputStream("fileNamePath.properties");

prop.load(fis);

} catch (FileNotFoundException e) {

e.getMessage();

} catch (IOException e) {

e.getMessage();

} catch (Exception e) {

e.getMessage();

}

System.out.println("Running script after exception");

}

<https://stackabuse.com/exception-handling-in-java-a-complete-guide-with-best-and-worst-practices/>

1. **What is the difference between throw and throws?**

**Throws :**

• is used to declare an exception, which means it works similar to the try-catch block.

• is used in method declaration.

• is followed by exception class names.

• you can declare multiple exception with throws

• throws declare at method it might throws Exception

• used to handover the responsibility of handling the exception occurred in the method to the

caller method.

**1 Example:**

public void readPropFile() throws FileNotFoundException, IOException{

Properties prop=new Properties();

FileInputStream fis=new FileInputStream("fileNamePath.properties");

prop.load(fis);

}

**2 Example:**

public class Test {

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

Test test = new Test();

test.company();

}

void studentDetails() throws InterruptedException {

System.out.println("School is over");

Thread.sleep(3000);

System.out.println("Please do not disturb.....");

}

void tsa() throws InterruptedException {

studentDetails();

}

void company() throws InterruptedException {

tsa();

}

}

**Throw :**

• is used in the method body to throw an exception

• throw is followed by an instance variable

• you cannot declare multiple exceptions with throw

• throw use for throwing actual Exception

• The throw keyword is used to handover the instance of the exception created by the

programmer to the JVM manually.

• throw keyword uses the exception object

**Example**

class Test {

public static void main(String args[]) {

classAge(15);

}

public static void classAge(int age) {

try {

if (age > 18) {

System.out.println("You are eligible to enroll in class");

}else {

System.out.println("You are not eligible to enroll in class");

}throw new ArithmeticException("student is less than 18 years old");

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}

<https://beginnersbook.com/2013/04/difference-between-throw-and-throws-in-java/>

1. **What is the difference between interface and a class? Example from your framework?**

**Class :**

• Class will contain concrete methods

• Class is extended

• We can create an Object of the class

• Class can inherit only one Class and can implement many interfaces

**Interface :**

• Interface will have Interface keyword.

• Interface will contain only abstract methods

• We cannot create object of interface

• Interface needs to be implemented

• Class can extends many interfaces

• We need to provide implementation to all methods when we implement interface to the class

public interface Insurances {

public void auto();

public void rental();

public void travel();

}

Interface is like a contract with the class, which says I only accept stuff, which looks like the signature

that I provide. And class that implements the interface says, sure I will make sure that I look like that.

**Practical Example:**

Basic statement we all know in Selenium is

WebDriver driver = new FirefoxDriver();

WebDriver itself is an Interface.

We are initializing Firefox browser using Selenium WebDriver. It means we are creating a reference

variable (driver) of the interface (WebDriver) and creating an Object.

Here WebDriver is an Interface and FirefoxDriver is a class.

<https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/>

1. **Difference between abstract class and interface?**

Abstract class:

• Abstract class will have abstract keyword.

• Abstract class can have both abstract as well as concrete methods

• Abstract methods in the sense no implementation only declaration.

• We cannot create an Object of Abstract Class

public abstract class Insurance {

public void auto() {

System.out.println("Insurance auto");

}

public void rental() {

System.out.println("Insurance rental");

}

//abstract method

public abstract void pet()

}

<https://javapapers.com/core-java/abstract-and-interface-core-java-2/difference-between-a-java-interface-and-a-java-abstract-class/>

1. **When to use abstract class and interface in Java?**

• An abstract class is good if you think you will plan on using inheritance since it provides a

common base class implementation to derived classes.

• An abstract class is also good if you want to be able to declare non-public members. In an

interface, all methods must be public.

• If you think you will need to add methods in the future, then an abstract class is a better choice.

Because if you add new method headings to an interface, then all of the classes that already

implement that interface will have to be changed to implement the new methods. That can be

quite a hassle.

<https://www.journaldev.com/1607/difference-between-abstract-class-and-interface-in-java>

1. **Explain OOPS concepts?**

OOP concepts in Java are the main ideas behind Java’s Object Oriented Programming. They are an

abstraction, encapsulation, inheritance, and polymorphism.

<https://stackify.com/oops-concepts-in-java/>

1. **What is polymorphism? Types of polymorphism? How do you use it in your code?**

Polymorphism is the ability of an object to take on many forms.

Combination of overloading and overriding is known as Polymorphism.

Method Overloading is called as Static Polymorphism/Compile time polymorphisms.

Method Overriding is also called Dynamic Polymorphism/Run time polymorphism.

<https://www.sitepoint.com/quick-guide-to-polymorphism-in-java/>

1. **Can we override/overload main method? Explain the reason? Can you override static method?**

We cannot override static method, so we cannot override main method.

However, you can overload main method in Java. But the program doesn't execute the overloaded

main method when you run your program, you have to call the overloaded main method from the

actual main method.

Practically I do not see any use of it and we don’t use it in my framework.

public class MainMethodOverload {

public static void main(String[] args) {

main(5);//if comment this line nothing will get executed

}

public static void main(int r) {

System.out.println("Hello");

}

}

<http://www.java67.com/2015/06/can-you-overload-or-override-main-in-java.html>

1. **How does method override differ from abstraction and inheritance?**

**Override**

When methods with same name and arguments present in child and parents class and class inheriting

the method from its **superclass** (parent class) has the option to override it. Benefit of overriding is the

ability to define behaviour specific to particular class. Method Overriding is possible only by

inheritance.

**Inheritance** is a process where one class inherits the properties of another class or simply we cans

say that extending one class into other class is known as **Inheritance.**

**Abstraction** is the methodology of hiding the implementation of internal details and showing the

functionality to the users.

<https://stackoverflow.com/questions/6308178/what-is-the-main-difference-between-inheritance-and-polymorphism>

1. **Method overloading & overriding? How do you use it in your framework?**

**Method overloading** in Java occurs when two or more methods in the same class have the exact same

name but different parameters (remember that method parameters accept values passed into the

method).

**Overloading:** Same method name with different arguments **in same class**

**Example**

public static void dropdown(WebElement Object,int value){

Select index= new Select(Object);

index.selectByIndex(value);

}

public static void dropdown(WebElement Object,String value){

Select visibletext= new Select(Object);

visibletext.selectByValue(value);

}

public static void dropdown(WebElement Object,Object value){

Select visibletext= new Select(Object);

visibletext.selectByVisibleText((String) value);

}

**An overridden method** would have the exact same method name, return type, number of parameters,

and types of parameters as the method in the parent class, and the only difference would be the body

of the method.

**Overriding**: Changing the implementation of the inherited methods **in the subclass(child class)**

With method overriding a child class can give its own specific implementation to an inherited method

without modifying the parent class method.

Combination of overloading and overriding is known as Polymorphism.

<https://softwaretestingboard.com/q2a/2915/provide-example-overloading-overriding-selenium-framework#axzz5orLva55a>

1. **What is singleton and have used singleton concept in your project ?**

I know what is singleton class, however in my current project I don not use this concept.

(A singleton class is a class that can have only one object (an instance of the class) at a time.)

<https://stackoverflow.com/questions/3192095/where-exactly-the-singleton-pattern-is-used-in-real-application>

1. **Can we achieve 100% abstraction in JAVA? Can we achieve 100% abstraction in JAVA**

**with use of the interfaces?**

We cannot achieve 100% abstraction in JAVA unless we use Interfaces

<https://stackoverflow.com/questions/34954592/how-can-interface-achieve-100-abstraction-in-java>

1. **What is encapsulation?**

It is the technique of making the fields in a class private and providing access to the fields via public

methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby

hiding the fields within the class. Therefore encapsulation is also referred to as data hiding.

The main benefit of encapsulation is the ability to modify our implemented code without breaking the

code of others who use our code. With this Encapsulation gives maintainability, flexibility and

extensibility to our code.

<https://www.tutorialspoint.com/java/java_encapsulation.htm>

1. **What is the Difference between final, finally?**

**Final keyword:**

• Used to declare constant values. The variable declared as final should be initialized

only once and cannot be changed.

• Used to prevent inheritance. Java classes declared as final cannot be extended.

• Used to prevent method overriding. Methods declared as final cannot be overridden.

**Example 1:**

final int b=30;

b=37; //cannot change value of final variable

**Example 2:**

public final class Test {

public static void main(String args[]) {

System.out.println("I am parent");

}

//you will get an error “Cannot subclass final class”

public class Child extends Test{

public static void main(String[] args) {

System.out.println("I am a child");

}

}

**Example 3:**

public class Test {

public final void testFinalKey() {

System.out.println("Parent final method");

}

}

//you will get an error “Cannot override the final method”

public class Child extends Test{

public void testFinalKey() {

System.out.println("Child final method");

}

}

**Finally block :**

• The finally block **always** executes when the try block exits. This ensures that the

finally block is executed even if an unexpected exception occurs.

try {

Properties prop = new Properties();

FileInputStream fis = new FileInputStream("FilePath");

prop.load(fis);

} catch (Exception e) {

System.out.println("I am an exception block");

} finally {

System.out.println("I am final block");

}

System.out.println("Running script after exception");

<https://www.javatpoint.com/difference-between-final-finally-and-finalize>

1. **What is Access Modifiers (Private, public, protected)? How did you use them?**

Java provides access modifiers to set access levels for classes, variables, methods and constructors.

**public:** A class or interface may be accessed from outside the package. Constructors, inner classes,

methods and field variables may be accessed wherever their class is accessed.

**protected:** Accessed by other classes in the same package or any subclasses of same package or

different package.

**private:** Accessed only within the class in which they are declared.

**default:** When no access modifier is specified for a class , method or data member – It is said to be

having the default access modifier by default.

<https://www.tutorialspoint.com/java/java_access_modifiers.htm>

1. **What is the output for this program?**

for (int i = 0; i < 3; i++) {

for (int j = 3; j >= 0; j--) {

if (i == j)

continue;

System.out.println(i + " " + j);

}

}

The continue keyword can be used in any of the loop control structures. It causes the loop to

immediately jump to the next iteration of the loop. So the output of the program will be:

0 3

0 2

0 1

1 3

1 2

1 0

2 3

2 1

2 0

1. **Here is the arrayList, how can I remove all duplicates from it?**

List<String> al = new ArrayList<String>();

al.add("Ajay");

al.add("Becky");

al.add("Chaitanya");

al.add("Ajay");

al.add("Rock");

al.add("Becky");

HashSet hs=new HashSet();

for (int i=0; i<al.size(); i++) {

hs.add(al.get(i));

}

System.out.println(hs);

<https://www.geeksforgeeks.org/how-to-remove-duplicates-from-arraylist-in-java/>

1. **What is the output of the following program?**

class Parent{

m1(){

System.out.println("In parent class m1");

}

}

class Subclass extends parent{

m1(){

System.out.println("In child class m1");

}

m2(){

System.out.println("In m2");

}

public static void main(String args[]){

Parent obj= new Subclass();

obj.m1();

obj.m2();

}

}

Program won’t run !

public static void main(String args[])

Parent obj= new Subclass();

obj.m1(); - will give child output ("In child class m1")

obj.m2(); - this method won’t be accessible (child class object is referred by parent

class reference variable)