1. What are the important features of Java 10 release?

Java 10 is the first every-six-months from Oracle corporation, so it's not a major release like earlier versions. However some of the important features of Java 10 are:

- Local-Variable Type Inference
- Enhance java.util.Locale and related APIs to implement additional Unicode extensions of BCP 47 language tags.
- Enable the HotSpot VM to allocate the Java object heap on an alternative memory device, such as an NV-DIMM, specified by the user.
- Provide a default set of root Certification Authority (CA) certificates in the JDK.

Java 10 is mostly a maintenance release, however I really liked the local variable type inference feature.

2. What are the important features of Java 9 release?

Java 9 was a major release and brought a lot of features. Some of the important features are:

- Java 9 REPL (JShell)
- Java 9 Module System
- Factory Methods for Immutable List, Set, Map and Map.Entry
- Private methods in Interfaces
- Reactive Streams
- GC (Garbage Collector) Improvements

You will find more details about them at Java 9 Features.

3. What are the important features of Java 8 release?

Java 8 has been released in March 2014, so it's one of the hot topic in java interview questions. If you answer this question clearly, it will show that you like to keep yourself up-to-date with the latest technologies.

Java 8 has been one of the biggest release after Java 5 annotations and generics. Some of the important features of Java 8 are:

1. Interface changes with default and static methods
2. Functional interfaces and Lambda Expressions
3. Java Stream API for collection classes
4. Java Date Time API

I strongly recommend to go through above links to get proper understanding of each one of them, also read Java 8 Features.

4. Name some OOPS Concepts in Java?

Java is based on Object Oriented Programming Concepts, following are some of the OOPS concepts implemented in java programming.

- Abstraction
- Encapsulation
- Polymorphism
- Inheritance
- Association

- Aggregation
- Composition

Read more about them at OOPS Concepts in Java.

5. What do you mean by platform independence of Java?

Platform independence means that you can run the same Java Program in any Operating System. For example, you can write java program in Windows and run it in Mac OS.

6. What is JVM and is it platform independent?

Java Virtual Machine (JVM) is the heart of java programming language. JVM is responsible for converting byte code into machine readable code. JVM is not platform independent, thats why you have different JVM for different operating systems. We can customize JVM with Java Options, such as allocating minimum and maximum memory to JVM. It's called virtual because it provides an interface that doesn't depend on the underlying OS.

7. What is the difference between JDK and JVM?

Java Development Kit (JDK) is for development purpose and JVM is a part of it to execute the java programs.

JDK provides all the tools, executables and binaries required to compile, debug and execute a Java Program. The execution part is handled by JVM to provide machine independence.

8. What is the difference between JVM and JRE?

Java Runtime Environment (JRE) is the implementation of JVM. JRE consists of JVM and java binaries and other classes to execute any program successfully. JRE doesn't contain any development tools like java compiler, debugger etc. If you want to execute any java program, you should have JRE installed.

9. Which class is the superclass of all classes?

`java.lang.Object` is the root class for all the java classes and we don't need to extend it.

10. Why Java doesn't support multiple inheritance?

Java doesn't support multiple inheritance in classes because of "Diamond Problem". To know more about diamond problem with example, read Multiple Inheritance in Java.

However multiple inheritances are supported in interfaces. An interface can extend multiple interfaces because they just declare the methods and implementation will be present in the implementing class. So there is no issue of the diamond problem with interfaces.

11. Why Java is not pure Object Oriented language?

Java is not said to be pure object-oriented because it supports primitive types such as int, byte, short, long etc. I believe it brings simplicity to the language while writing our code. Obviously, java could have wrapper objects for the primitive types but just for the representation, they would not have provided any benefit.

As we know, for all the primitive types we have wrapper classes such as Integer, Long etc that provides some additional methods.

12. What is difference between path and classpath variables?

PATH is an environment variable used by operating system to locate the executables. That's why when we install Java or want any executable to be found by OS, we need to add the directory location in the PATH variable. If you work on Windows OS, read this post to learn how to setup PATH variable on Windows.

Classpath is specific to java and used by java executables to locate class files. We can provide the classpath location while running java application and it can be a directory, ZIP files, JAR files etc.

13. What is the importance of main method in Java?

main() method is the entry point of any standalone java application. The syntax of main method is `public static void main(String args[])`.

Java main method is public and static so that Java runtime can access it without initializing the class. The input parameter is an array of String through which we can pass runtime arguments to the java program. Check this post to learn how to compile and run java program.

14. What is overloading and overriding in java?

When we have more than one method with the same name in a single class but the arguments are different, then it is called as method overloading.

Overriding concept comes in picture with inheritance when we have two methods with same signature, one in parent class and another in child class. We can use @Override annotation in the child class overridden method to make sure if parent class method is changed, so as child class.

15. Can we overload main method?

Yes, we can have multiple methods with name "main" in a single class. However if we run the class, java runtime environment will look for main method with syntax as `public static void main(String args[])`.

16. Can we have multiple public classes in a java source file?

We can't have more than one public class in a single java source file. A single source file can have multiple classes that are not public.

17. What is Java Package and which package is imported by default?

Java package is the mechanism to organize the java classes by grouping them. The grouping logic can be based on functionality or modules based. A java class fully classified name contains package and class name. For example, `java.lang.Object` is the fully classified name of `Object` class that is part of `java.lang` package.

`java.lang` package is imported by default and we don't need to import any class from this package explicitly.

18. What are access modifiers?

Java provides access control through public, private and protected access modifier keywords. When none of these are used, it's called default access modifier.
A java class can only have public or default access modifier. Read Java Access Modifiers to learn more about these in detail.

19. What is final keyword?

final keyword is used with Class to make sure no other class can extend it, for example String class is final and we can't extend it.

We can use the final keyword with methods to make sure child classes can't override it.

final keyword can be used with variables to make sure that it can be assigned only once. However the state of the variable can be changed, for example, we can assign a final variable to an object only once but the object variables can change later on.

Java interface variables are by default final and static.

20. What is static keyword?

static keyword can be used with class level variables to make it global i.e all the objects will share the same variable.

static keyword can be used with methods also. A static method can access only static variables of class and invoke only static methods of the class.

Read more in detail at java static keyword.

21. What is finally and finalize in java?

finally block is used with try-catch to put the code that you want to get executed always, even if any exception is thrown by the try-catch block. finally block is mostly used to release resources created in the try block.

finalize() is a special method in Object class that we can override in our classes. This method gets called by the garbage collector when the object is getting garbage collected. This method is usually overridden to release system resources when the object is garbage collected.

22. Can we declare a class as static?

We can't declare a top-level class as static however an inner class can be declared as static. If inner class is declared as static, it's called static nested class.
The static nested class is same as any other top-level class and is nested for only packaging convenience.

Read more about inner classes at java inner class.

23. What is static import?

If we have to use any static variable or method from other class, usually we import the class and then use the method/variable with class name.

```
import java.lang.Math;

//inside class
double test = Math.PI * 5;
```

We can do the same thing by importing the static method or variable only and then use it in the class as if it belongs to it.

```
import static java.lang.Math.PI;
```

```
//no need to refer class now
double test = PI * 5;
```

Use of static import can cause confusion, so it's better to avoid it. Overuse of static import can make your program unreadable and unmaintainable.

24. What is try-with-resources in java?

One of the Java 7 features is the try-with-resources statement for automatic resource management. Before Java 7, there was no auto resource management and we should explicitly close the resource. Usually, it was done in the `finally`block of a try-catch statement. This approach used to cause memory leaks when we forgot to close the resource.

From Java 7, we can create resources inside try block and use it. Java takes care of closing it as soon as try-catch block gets finished. Read more at Java Automatic Resource Management.

25. What is multi-catch block in java?

Java 7 one of the improvement was multi-catch block where we can catch multiple exceptions in a single catch block. This makes are code shorter and cleaner when every catch block has similar code.

If a catch block handles multiple exceptions, you can separate them using a pipe (|) and in this case, exception parameter (ex) is final, so you can't change it.

Read more at Java multi catch block.

26. What is static block?

Java static block is the group of statements that gets executed when the class is loaded into memory by Java ClassLoader. It is used to initialize static variables of the class. Mostly it's used to create static resources when class is loaded.

27. What is an interface?

Interfaces are core part of java programming language and used a lot not only in JDK but also java design patterns, most of the frameworks and tools. Interfaces provide a way to achieve abstraction in java and used to define the contract for the subclasses to implement.

Interfaces are good for starting point to define Type and create top level hierarchy in our code. Since a java class can implements multiple interfaces, it's better to use interfaces as super class in most of the cases. Read more at java interface.

28. What is an abstract class?

Abstract classes are used in java to create a class with some default method implementation for subclasses. An abstract class can have abstract method without body and it can have methods with implementation also.

abstract keyword is used to create a abstract class. Abstract classes can't be instantiated and mostly used to provide base for sub-classes to extend and implement the abstract methods and override or use

the implemented methods in abstract class. Read important points about abstract classes at java abstract class.

29. What is the difference between abstract class and interface?

abstract keyword is used to create abstract class whereas interface is the keyword for interfaces.

Abstract classes can have method implementations whereas interfaces can't.

A class can extend only one abstract class but it can implement multiple interfaces.

We can run abstract class if it has main() method whereas we can't run an interface.

Some more differences in detail are at Difference between Abstract Class and Interface.

30. Can an interface implement or extend another interface?

Interfaces don't implement another interface, they extend it. Since interfaces can't have method implementations, there is no issue of diamond problem. That's why we have multiple inheritance in interfaces i.e an interface can extend multiple interfaces.

From Java 8 onwards, interfaces can have default method implementations. So to handle diamond problem when a common default method is present in multiple interfaces, it's mandatory to provide implementation of the method in the class implementing them. For more details with examples, read Java 8 interface changes.

31. What is Marker interface?

A marker interface is an empty interface without any method but used to force some functionality in implementing classes by Java. Some of the well known marker interfaces are Serializable and Cloneable.

32. What are Wrapper classes?

Java wrapper classes are the Object representation of eight primitive types in java. All the wrapper classes in java are immutable and final. Java 5 autoboxing and unboxing allows easy conversion between primitive types and their corresponding wrapper classes.

Read more at Wrapper classes in Java.

33. What is Enum in Java?

Enum was introduced in Java 1.5 as a new type whose fields consists of fixed set of constants. For example, in Java we can create Direction as enum with fixed fields as EAST, WEST, NORTH, SOUTH.

enum is the keyword to create an enum type and similar to the class. Enum constants are implicitly static and final. Read more in detail at java enum.

34. What is Java Annotations?

Java Annotations provide information about the code and they have no direct effect on the code they annotate. Annotations are introduced in Java 5. Annotation is metadata about the program embedded in the program itself. It can be parsed by the annotation parsing tool or by the compiler. We can also specify annotation availability to either compile time only or till runtime also. Java Built-in annotations are @Override, @Deprecated and @SuppressWarnings. Read more at java annotations.

35. What is Java Reflection API? Why it's so important to have?

Java Reflection API provides the ability to inspect and modify the runtime behavior of java application. We can inspect a java class, interface, enum and get their methods and field details. Reflection API is an advanced topic and we should avoid it in normal programming. Reflection API usage can break the design pattern such as Singleton pattern by invoking the private constructor i.e violating the rules of access modifiers.

Even though we don't use Reflection API in normal programming, it's very important to have. We can't have any frameworks such as Spring, Hibernate or servers such as Tomcat, JBoss without Reflection API. They invoke the appropriate methods and instantiate classes through reflection API and use it a lot for other processing.

Read Java Reflection Tutorial to get in-depth knowledge of reflection api.

36. What is composition in java?

Composition is the design technique to implement has-a relationship in classes. We can use Object composition for code reuse.

Java composition is achieved by using instance variables that refer to other objects. The benefit of using composition is that we can control the visibility of other objects to client classes and reuse only what we need. Read more with example at Java Composition example.

37. What is the benefit of Composition over Inheritance?

One of the best practices of Java programming is to "favor composition over inheritance". Some of the possible reasons are:

- Any change in the superclass might affect subclass even though we might not be using the superclass methods. For example, if we have a method test() in the subclass and suddenly somebody introduces a method test() in the superclass, we will get compilation errors in the subclass. Composition will never face this issue because we are using only what methods we need.
- Inheritance exposes all the superclass methods and variables to the client and if we have no control in designing superclass, it can lead to security holes. Composition allows us to provide restricted access to the methods and hence more secure.
- We can get runtime binding in composition where inheritance binds the classes at compile time. So composition provides flexibility in the invocation of methods.

You can read more about above benefits of composition over inheritance at java composition vs inheritance.

38. How to sort a collection of custom Objects in Java?

We need to implement Comparable interface to support sorting of custom objects in a collection. Comparable interface has compareTo(T obj) method which is used by sorting methods and by providing this method implementation, we can provide default way to sort custom objects collection.

However, if you want to sort based on different criteria, such as sorting an Employees collection based on salary or age, then we can create Comparator instances and pass it as sorting methodology. For more details read Java Comparable and Comparator.

39. What is inner class in java?

We can define a class inside a class and they are called nested classes. Any non-static nested class is known as inner class. Inner classes are associated with the object of the class and they can access all the variables and methods of the outer class. Since inner classes are associated with the instance, we can't have any static variables in them.

We can have local inner class or anonymous inner class inside a class. For more details read java inner class.

40. What is anonymous inner class?

A local inner class without name is known as anonymous inner class. An anonymous class is defined and instantiated in a single statement. Anonymous inner class always extend a class or implement an interface.

Since an anonymous class has no name, it is not possible to define a constructor for an anonymous class. Anonymous inner classes are accessible only at the point where it is defined.

41. What is Classloader in Java?

Java Classloader is the program that loads byte code program into memory when we want to access any class. We can create our own classloader by extending ClassLoader class and overriding loadClass(String name) method. Learn more at java classloader.

42. What are different types of classloaders?

There are three types of built-in Class Loaders in Java:

1. Bootstrap Class Loader – It loads JDK internal classes, typically loads rt.jar and other core classes.
2. Extensions Class Loader – It loads classes from the JDK extensions directory, usually $JAVA_HOME/lib/ext directory.
3. System Class Loader – It loads classes from the current classpath that can be set while invoking a program using -cp or -classpath command line options.

43. What is ternary operator in java?

Java ternary operator is the only conditional operator that takes three operands. It's a one liner replacement for if-then-else statement and used a lot in java programming. We can use ternary operator if-else conditions or even switch conditions using nested ternary operators. An example can be found at java ternary operator.

44. What does super keyword do?

super keyword can be used to access super class method when you have overridden the method in the child class.

We can use super keyword to invoke superclass constructor in child class constructor but in this case, it should be the first statement in the constructor method.

```
package com.journaldev.access;

public class SuperClass {

	public SuperClass(){
	}

	public SuperClass(int i){}

	public void test(){
		System.out.println("super class test method");
	}
}
```

Use of super keyword can be seen in below child class implementation.

```
package com.journaldev.access;

public class ChildClass extends SuperClass {

	public ChildClass(String str){
		//access super class constructor with super keyword
		super();

		//access child class method
		test();

		//use super to access super class method
		super.test();
	}

	@Override
	public void test(){
		System.out.println("child class test method");
	}
}
```

45. What is break and continue statement?

We can use break statement to terminate for, while, or do-while loop. We can use break statement in switch statement to exit the switch case. You can see the example of break statement at java break. We can use break with label to terminate the nested loops.

The continue statement skips the current iteration of a for, while or do-while loop. We can use continue statement with the label to skip the current iteration of the outermost loop.

46. What is this keyword?

this keyword provides the reference to the current object and it's mostly used to make sure that object variables are used, not the local variables having the same name.

```
//constructor
public Point(int x, int y) {
    this.x = x;
    this.y = y;
}
```

We can also use this keyword to invoke other constructors from a constructor.

```
public Rectangle() {
    this(0, 0, 0, 0);
}
public Rectangle(int width, int height) {
    this(0, 0, width, height);
}
public Rectangle(int x, int y, int width, int height) {
    this.x = x;
    this.y = y;
```

```
        this.width = width;
        this.height = height;
    }
```

47. What is default constructor?

No argument constructor of a class is known as default constructor. When we don't define any constructor for the class, java compiler automatically creates the default no-args constructor for the class. If there are other constructors defined, then compiler won't create default constructor for us.

48. Can we have try without catch block?

Yes, we can have try-finally statement and hence avoiding catch block.

49. What is Garbage Collection?

Garbage Collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. In Java, process of deallocating memory is handled automatically by the garbage collector.

We can run the garbage collector with code `Runtime.getRuntime().gc()` or use utility method `System.gc()`. For a detailed analysis of Heap Memory and Garbage Collection, please read Java Garbage Collection.

50. What is Serialization and Deserialization?

We can convert a Java object to an Stream that is called Serialization. Once an object is converted to Stream, it can be saved to file or send over the network or used in socket connections.

The object should implement Serializable interface and we can use java.io.ObjectOutputStream to write object to file or to any OutputStream object. Read more at Java Serialization.

The process of converting stream data created through serialization to Object is called deserialization. Read more at Java Deserialization.

51. How to run a JAR file through command prompt?

We can run a jar file using java command but it requires Main-Class entry in jar manifest file. Main-Class is the entry point of the jar and used by java command to execute the class. Learn more at java jar file.

52. What is the use of System class?

Java System Class is one of the core classes. One of the easiest way to log information for debugging is System.out.print() method.

System class is final so that we can't subclass and override its behavior through inheritance. System class doesn't provide any public constructors, so we can't instantiate this class and that's why all of its methods are static.

Some of the utility methods of System class are for array copy, get the current time, reading environment variables. Read more at Java System Class.

53. What is instanceof keyword?

We can use instanceof keyword to check if an object belongs to a class or not. We should avoid it's usage as much as possible. Sample usage is:

```
public static void main(String args[]){
        Object str = new String("abc");

        if(str instanceof String){
                System.out.println("String value:"+str);
        }

        if(str instanceof Integer){
                System.out.println("Integer value:"+str);
        }
}
```

Since str is of type String at runtime, first if statement evaluates to true and second one to false.

54. Can we use String with switch case?

One of the Java 7 feature was improvement of switch case of allow Strings. So if you are using Java 7 or higher version, you can use String in switch-case statements. Read more at Java switch-case String example.

55. Java is Pass by Value or Pass by Reference?

This is a very confusing question, we know that object variables contain the reference to the Objects in heap space. When we invoke any method, a copy of these variables is passed and gets stored in the stack memory of the method. We can test any language whether it's pass by reference or pass by value through a simple generic swap method, to learn more read Java is Pass by Value and Not Pass by Reference.

56. What is difference between Heap and Stack Memory?

Major difference between Heap and Stack memory are as follows:

- Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.
- Whenever an object is created, it's always stored in the Heap space and stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
- Memory management in the stack is done in LIFO manner whereas it's more complex in Heap memory because it's used globally.

For a detailed explanation with a sample program, read Java Heap vs Stack Memory.

57. Java Compiler is stored in JDK, JRE or JVM?

The task of java compiler is to convert java program into bytecode, we have `javac` executable for that. So it must be stored in JDK, we don't need it in JRE and JVM is just the specs.

58. What will be the output of following programs?

1. **static method in class**

```
package com.journaldev.util;
```

```
public class Test {

	public static String toString(){
		System.out.println("Test toString called");
		return "";
	}

	public static void main(String args[]){
		System.out.println(toString());
	}
}
```

Answer: The code won't compile because we can't have an `Object` class method with static keyword. Note that Object class has `toString()`method. You will get compile time error as "This static method cannot hide the instance method from Object". The reason is that static method belongs to the class and since every class base is Object, we can't have the same method in the instance as well as in class. You won't get this error if you change the method name from toString() to something else that is not present in superclass `Object`.

2. **static method invocation**

```
package com.journaldev.util;

public class Test {

	public static String foo(){
		System.out.println("Test foo called");
		return "";
	}

	public static void main(String args[]){
		Test obj = null;
		System.out.println(obj.foo());
	}
}
```

Answer: Well this is a strange situation. We all have seen `NullPointerException` when we invoke a method on the object that is NULL. But here this program will work and prints "Test foo called".

The reason for this is the java compiler code optimization. When the java code is compiled to produced byte code, it figures out that foo() is a static method and should be called using class. So it changes the method call `obj.foo()` to `Test.foo()` and hence no `NullPointerException`.