**Java Interview Questions**

**Basic Java Interview Questions**

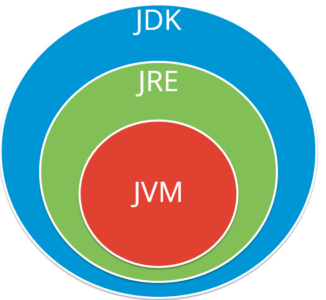
**Q1. Explain JVM, JRE and JDK?**

**JVM (Java Virtual Machine):**It is an abstract machine. It is a specification that provides run-time environment in which java bytecode can be executed. It follows three notations:

* **Specification**: It is a document that describes the implementation of the Java virtual machine. It is provided by Sun and other companies.
* **Implementation**: It is a program that meets the requirements of JVM specification.
* **Runtime Instance**: An instance of JVM is created whenever you write a java command on the command prompt and run the class.

**JRE (Java Runtime Environment) :**JRE refers to a runtime environment in which java bytecode can be executed. It implements the JVM (Java Virtual Machine) and provides all the class libraries and other support files that JVM uses at runtime. So JRE is a software package that contains what is required to run a Java program. Basically, it’s an implementation of the JVM which physically exists.

**JDK(Java Development Kit) :**It is the tool necessary to compile, document and package Java programs. The JDK completely includes JRE which contains tools for Java programmers. The Java Development Kit is provided free of charge. Along with JRE, it includes an interpreter/loader, a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development. In short, it contains JRE + development tools.



**Static Keyword**

The static keyword denotes that a member variable, or method, can be accessed without requiring an instantiation of the class to which it belongs.

In simple terms, it means that you can call a method, even if you've never created the object to which it belongs! Every time you run a stand-alone application (which requires a static main method), the virtual machine can call the main method without creating a new application object. Of course, unless the application's methods are all static, you will need to create an instance of it at some point.

**1) Java static variable**

* If you declare any variable as static, it is known static variable.
* The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
* The static variable gets memory only once in class area at the time of class loading.

**2) Java Static Method**

* If you apply static keyword with any method, it is known as static method.
* A static method belongs to the class rather than object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* static method can access static data member and can change the value of it.

**3) Java static block**

* Is used to initialize the static data member.
* It is executed before main method at the time of classloading.

1. class A2{
2. static{System.out.println("static block is invoked");}
3. public static void main(String args[]){
4. System.out.println("Hello main");
5. }
6. }

**Q2. Explain public static void main(String args[]).**

**public**: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.

**static** : It is a keyword in java which identifies it is class based i.e it can be accessed without creating the instance of a Class.

**void** : It is the return type of the method. Void defines the method which will not return any value.

**main**: It is the name of the methodwhich is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.

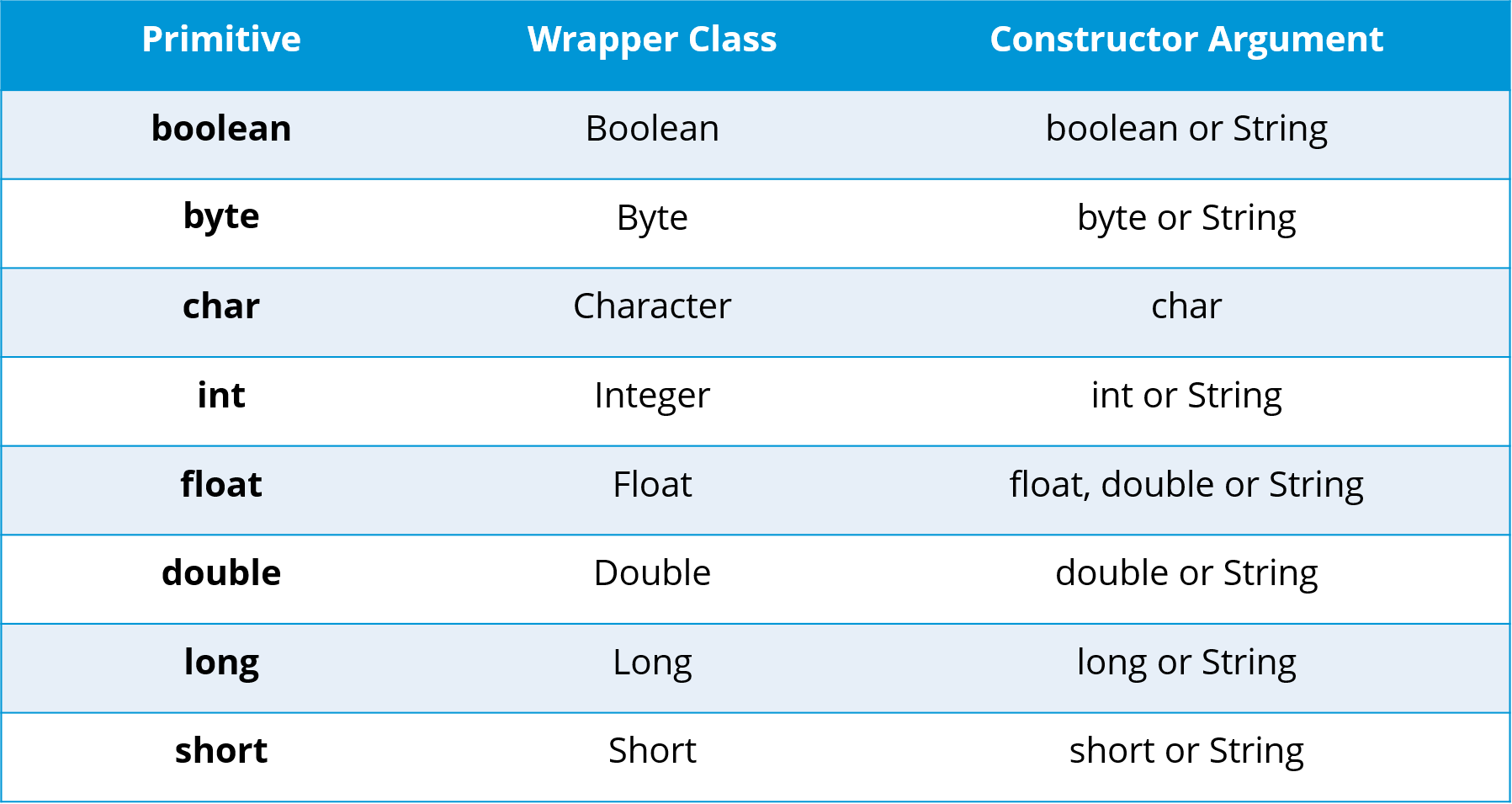
**String args[]** : It is the parameter passed to the main method.

**Q4. Why java is not 100% Object-oriented?**

Java is not 100% Object-oriented because it makes use of eight primitive datatypes such as boolean, byte, char, int, float, double, long, short which are **not objects.**

**Q5. What are wrapper classes?**

Wrapper classes converts the java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays different primitive type, wrapper class and constructor argument.



**Q6. What are constructors in Java?**

In Java, constructor refers to a block of code which is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

**Q7. What is singleton class and how can we make a class singleton?**

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

**Q8. What is the difference between Array list and vector?**

|  |  |
| --- | --- |
| **Array List** | **Vector** |
| Array List is not synchronized. | Vector is synchronized. |
| Array List is fast as it’s non-synchronized. | Vector is slow as it is thread safe. |
| If an element is inserted into the Array List, it increases its Array size by 50%. | Vector defaults to doubling size of its array. |
| Array List does not define the increment size. | Vector defines the increment size. |
| Array List can only use Iterator for traversing an Array List. | Except Hashtable, Vector is the only other class which uses both Enumeration and Iterator. |

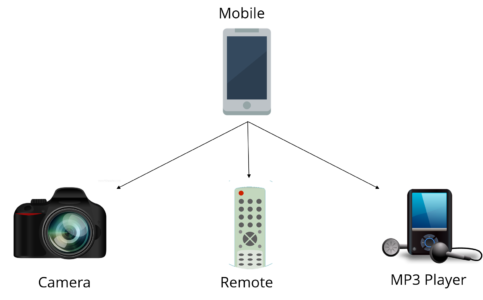
**Q9. What is the difference between equals() and == ?**

Equals() method is defined in Object class in Java and used for checking equality of two objects defined by business logic.  
“==” or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. *public boolean equals(Object o)* is the method provided by the Object class. The default implementation uses == operator to compare two objects. For example: method can be overridden like String class. equals() method is used to compare the values of two objects.

|  |  |
| --- | --- |
|  |  |
|  |  |

**OOPS Java Interview Questions:**

**Q1. What is Polymorphism?**



Polymorphism is briefly described as “one interface, many implementations”. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are two types of polymorphism:

1. Compile time polymorphism
2. Run time polymorphism

**Abstract Class**

Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods. Let's look at an example of an abstract class, and an abstract method.

Suppose we were modeling the behavior of animals, by creating a class hierachy that started with a base class called Animal. Animals are capable of doing different things like flying, digging and walking, but there are some common operations as well like eating and sleeping. Some common operations are performed by all animals, but in a different way as well. When an operation is performed in a different way, it is a good candidate for an abstract method (forcing subclasses to provide a custom implementation). Let's look at a very primitive Animal base class, which defines an abstract method for making a sound (such as a dog barking, a cow mooing, or a pig oinking).

**Q3. What is the difference between abstract classes and interfaces?**

|  |  |
| --- | --- |
| **Abstract Class** | **Interfaces** |
| An abstract class can provide complete, default code and/or just the details that have to be overridden. | An interface cannot provide any code at all,just the signature. |
| In case of abstract class, a class may extend only one abstract class. | A Class may implement several interfaces. |
| An abstract class can have non-abstract methods. | All methods of an Interface are abstract. |
| An abstract class can have instance variables. | An Interface cannot have instance variables |
| An abstract class can have any visibility: public, private, protected. | An Interface visibility must be public (or) none. |
| If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly | If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method |
| An abstract class can contain constructors | An Interface cannot contain constructors |
| Abstract classes are fast | Interfaces are slow as it requires extra indirection to find corresponding method in the actual class |

**Q4. What is method overloading and method overriding?**

**Method Overloading :**

* In Method Overloading, Methods of the same class shares the same name but each method must have different number of parameters or parameters having different types and order.
* Method Overloading is to “add” or “extend” more to method’s behavior.
* It is a compile time polymorphism.
* The methods must have different signature.
* It may or may not need inheritance in Method Overloading.

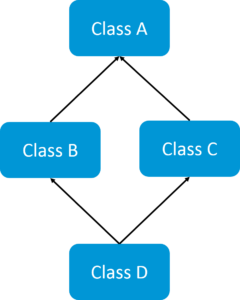
**Method Overriding:**

* In Method Overriding, sub class have the same method with same name and exactly the same number and type of parameters and same return type as a super class.
* Method Overriding is to “Change” existing behavior of method.
* It is a run time polymorphism.
* The methods must have same signature.
* It always requires inheritance in Method Overriding.

**Q5. Can you override a private or static method in Java?**

You cannot override a private or static method in Java. If you create a similar method with same return type and same method arguments in child class then it will hide the super class method; this is known as method hiding. Similarly, you cannot override a private method in sub class because it’s not accessible there. What you can do is create another private method with the same name in the child class. Let’s take a look at the example below to understand it better.

**Q6. What is multiple inheritance? Is it supported by Java?**

If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that if multiple parent classes have a same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.

Therefore, Java doesn’t support multiple inheritance. The problem is commonly referred as Diamond Problem.

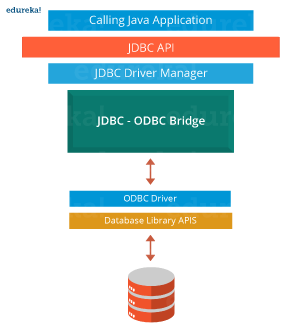
In case you are facing any challenges with these java interview questions, please comment your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for a structured training from edureka! Click below to know more.

**JDBC Interview Questions**

**Q1. What is JDBC Driver?**

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

****

**Q2. What are the steps to connect to a database in java?**

* Registering the driver class
* Creating connection
* Creating statement
* Executing queries
* Closing connection

**Q3. What are the JDBC API components?**

The java.sql package contains interfaces and classes for JDBC API.

**Interfaces:**

* Connection
* Statement
* PreparedStatement
* ResultSet
* ResultSetMetaData
* DatabaseMetaData
* CallableStatement etc.

**Classes:**

* DriverManager
* Blob
* Clob
* Types
* SQLException etc.

**Spring Interview Questions**

**Q1. What is a Spring?**

Wikipedia defines the Spring framework as “an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.” Spring is essentially a lightweight, integrated framework that can be used for developing enterprise applications in java.

**Q2. Name the different modules of the Spring framework.**

Some of the important Spring Framework modules are:

* Spring Context – for dependency injection.
* Spring AOP – for aspect oriented programming.
* Spring DAO – for database operations using DAO pattern
* Spring JDBC – for JDBC and DataSource support.
* Spring ORM – for ORM tools support such as Hibernate
* Spring Web Module – for creating web applications.
* Spring MVC – Model-View-Controller implementation for creating web applications, web services etc.

**Q3. List some of the important annotations in annotation-based Spring configuration.**

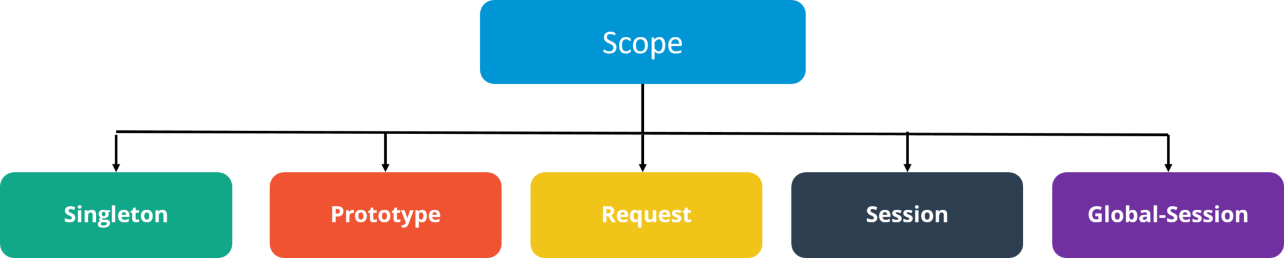
The important annotations are:

* @Required
* @Autowired
* @Qualifier
* @Resource
* @PostConstruct
* @PreDestroy

**Q4. Explain Bean in Spring and List the different Scopes of Spring bean.**

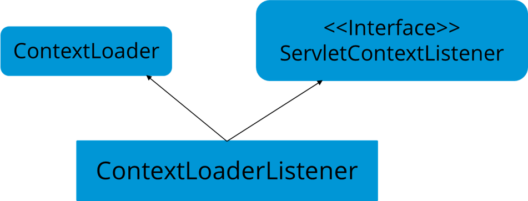
Beans are objects that form the backbone of a Spring application. They are managed by the Spring IoC container. In other words, a bean is an object that is instantiated, assembled, and managed by a Spring IoC container.

There are five Scopes defined in Spring beans.



* **Singleton**: Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues because it’s not thread-safe.
* **Prototype**: A new instance will be created every time the bean is requested.
* **Request**: This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
* **Session**: A new bean will be created for each HTTP session by the container.
* **Global-session**: This is used to create global session beans for Portlet applications.

**ContextLoaderListener,** on the other hand, is the listener to start up and shut down the WebApplicationContext in Spring root. Some of its important functions includes tying up the lifecycle of Application Context to the lifecycle of the ServletContext and automating the creation of ApplicationContext.

****

**Q8. How to handle exceptions in Spring MVC Framework?**

Spring MVC Framework provides following ways to help us achieving robust exception handling.

**Controller Based:**

We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.

**Global Exception Handler:**

Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.

**HandlerExceptionResolver implementation:**

For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

**Q9. What are some of the important Spring annotations which you have used?**

Some of the Spring annotations that I have used in my project are:

**@Controller** – for controller classes in Spring MVC project.

**@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples

**@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.

**@PathVariable** – for mapping dynamic values from the URI to handler method arguments.

**@Autowired** – for autowiring dependencies in spring beans.

**@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

**@Service** – for service classes.

**@Scope** – for configuring scope of the spring bean.

**@Configuration, @ComponentScan and @Bean** – for java based configurations.

AspectJ annotations for configuring aspects and advices, @Aspect, @Before, @After, @Around, @Pointcut etc.

**Q10. How to integrate Spring and Hibernate Frameworks?**

We can use Spring ORM module to integrate Spring and Hibernate frameworks, if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.

In case you are facing any challenges with these java interview questions, please comment your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for a structured training from edureka! Click below to know more.

**Exception and Thread Java Interview Questions**

**Q1. What is difference between Error and Exception?**

An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you can not repair them at runtime.Though error can be caught in catch block but the execution of application will come to a halt and is not recoverable.

While exceptions are conditions that occur because of bad input or human error etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving user a feedback for entering proper values etc.

**Q2. How can you handle Java exceptions?**

There are five keywords used to handle exceptions in java:

1. try
2. catch
3. finally
4. throw
5. throws

**Q3. What are the differences between Checked Exception and Unchecked Exception?**

**Checked Exception**

* The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions.
* Checked exceptions are checked at compile-time.
* Example: IOException, SQLException etc.

**Unchecked Exception**

* The classes that extend RuntimeException are known as unchecked exceptions.
* Unchecked exceptions are not checked at compile-time.
* Example: ArithmeticException, NullPointerException etc.

**Q4. What purpose does the keywords final, finally, and finalize fulfill?**

**Final:**

Final is used to apply restrictions on class, method and variable. Final class can’t be inherited, final method can’t be overridden and final variable value can’t be changed. Let’s take a look at the example below to understand it better.

|  |  |
| --- | --- |
|  |  |

**Finally**

Finally is used to place important code, it will be executed whether exception is handled or not. Let’s take a look at the example below to understand it better.

|  |  |
| --- | --- |
|  |  |

**Finalize**

Finalize is used to perform clean up processing just before object is garbage collected. Let’s take a look at the example below to understand it better.

|  |  |
| --- | --- |
|  |  |

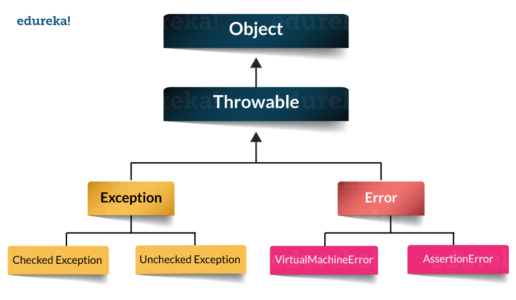
**Q5. What are the differences between throw and throws?**

|  |  |
| --- | --- |
| **throw keyword** | **throws keyword** |
| Throw is used to explicitly throw an exception. | Throws is used to declare an exception. |
| Checked exceptions can not be propagated with throw only. | Checked exception can be propagated with throws. |
| Throw is followed by an instance. | Throws is followed by class. |
| Throw is used within the method. | Throws is used with the method signature. |
| You cannot throw multiple exception | You can declare multiple exception e.g. public void method()throws IOException,SQLException. |

**Q6. What is exception hierarchy in java?**

The hierarchy is as follows:

Throwable is a parent class of all Exception classes. There are two types of Exceptions: Checked exceptions and UncheckedExceptions or RunTimeExceptions. Both type of exceptions extends Exception class whereas errors are further classified into Virtual Machine error and Assertion error.



**Q7. How to create a custom Exception?**

To create you own exception extend the Exception class or any of its subclasses.

* class New1Exception extends Exception { }               // this will create Checked Exception
* class NewException extends IOExcpetion { }             // this will create Checked exception
* class NewException extends NullPonterExcpetion { }  // this will create UnChecked exception

**Q8. What are the important methods of Java Exception Class?**

Exception and all of it’s subclasses doesn’t provide any specific methods and all of the methods are defined in the base class Throwable.

1. **String getMessage()** – This method returns the message String of Throwable and the message can be provided while creating the exception through it’s constructor.
2. **String getLocalizedMessage(**) – This method is provided so that subclasses can override it to provide locale specific message to the calling program. Throwable class implementation of this method simply use getMessage() method to return the exception message.
3. **Synchronized Throwable getCause()** – This method returns the cause of the exception or null id the cause is unknown.
4. **String toString()** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
5. **void printStackTrace()** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as argument to write the stack trace information to the file or stream.

**Q9. What are the differences between processes and threads?**

|  |  |  |
| --- | --- | --- |
|  | **Process** | **Thread** |
| **Definition** | An executing instance of a program is called a process. | A thread is a subset of the process. |
| **Communication** | Processes must use inter-process communication to communicate with sibling processes. | Threads can directly communicate with other threads of its process. |
| **Control** | Processes can only exercise control over child processes. | Threads can exercise considerable control over threads of the same process. |
| **Changes** | Any change in the parent process does not affect child processes. | Any change in the main thread may affect the behavior of the other threads of the process. |
| **Memory** | Run in separate memory spaces. | Run in shared memory spaces. |
| **Controlled by** | Process is controlled by the operating system. | Threads are controlled by programmer in a program. |
| **Dependence** | Processes are independent. | Threads are dependent. |

**Q10. What is a finally block? Is there a case when finally will not execute?**

Finally block is a block which always execute a set of statements. It is always associated with a try block regardless of any exception that occurs or not.   
Yes, finally will not be executed if the program exits either by calling System.exit() or by causing a fatal error that causes the process to abort.

**Q11. What is synchronization?**

Synchronization refers to multi-threading. A synchronized block of code can be executed by only one thread at a time. As Java supports execution of multiple threads, two or more threads may access the same fields or objects. Synchronization is a process which keeps all concurrent threads in execution to be in sync. Synchronization avoids memory consistency errors caused due to inconsistent view of shared memory. When a method is declared as synchronized the thread holds the monitor for that method’s object. If another thread is executing the synchronized method the thread is blocked until that thread releases the monitor.