**OBJECT ORIENTED PROGRAMMING**

**VIVA QUESTIONS WITH ANSWER**

**TOPIC:ENCAPSULATION IN JAVA**

**1.What is Encapsulation in Java? Why is it called Data hiding?**

Ans: The process of binding data and corresponding methods (behavior) together into a single unit is called encapsulation in Java.

In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from being accessed by other classes, thereby we can keep variables and methods safes from outside interference and misuse.

If a field is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within the class. Therefore, Encapsulation is also called data hiding.

2. **How to achieve encapsulation in Java? Give an example.**

Ans: There are two key points that should be kept in mind to achieve the encapsulation in Java. They are as follows:

* Declare the variable of the class as private.
* Provide public setter and getter methods to modify the values of variables.

3. **What are the main benefits of using encapsulation in Java?**

Ans: The main benefits of using encapsulation are:

* The main benefit of encapsulation is the ability to modify the implemented code without breaking the others code who have implemented the code.
* It also provides us with maintainability, flexibility, and extensibility to our code.
* The fields of a class can be made read-only or write-only.
* A class can have total control over what is stored in its fields.

4. **What is the difference between Abstraction and Encapsulation?**  
Or, how abstraction is different from encapsulation in Java?

Ans: There are the following differences between Abstraction and Encapsulation:

a) Abstraction solves the problem at the design level whereas encapsulation solves the problem at the implementation level.

b) Abstraction is implemented in Java using Interface and Abstract class whereas encapsulation is implemented using private and protected access modifiers.

c) Abstraction is used to hide the unwanted data and giving relevant data whereas encapsulation is used for hiding data and code in a single unit to prevent access from outside.

d) The real-time example of Abstraction is TV Remote Button whereas the real-time example of Encapsulation is medical medicine.

**5. Can we achieve abstraction without encapsulation in Java?**

Ans: Yes, we can achieve abstraction without encapsulation because both are different things and different concepts.

6.**How can the variable of the EncapsulationTest be accessed by using getter and setter methods?**

Ans: The public setXXX() and getXXX() methods are access points of the instance variable of EncapsulationTest class. Basically, these methods are known as getter and setter methods.

Therefore, any class that wants to access variable should access them through these getters and setters. The variables of the EncapsulationTest class can be accessed  as shown in the following code:



1

public class RunEncapTest {

2

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

3

EncaptulationTest encap = new EncapsulationTest();

4

  encap.setName("John");

5

  encap.setAge(22);

6

  encap.setId(123456);

7

​

8

System.out.println("Name: " +encap.getName()); System.out.println("Age: " +encap.getAge());

9

System.out.println("Id: " +encap.getId());

10

}

11

}

**7. What are getter and setter methods in Java?**

Ans: In Java, setter method is a method that is used for updating the values of a variable. This method is also known as mutator method.

Getter method is a method that is used to retrieve the value of a variable or return the value of the private member variable. This method is also known as an accessor method.

**Viva Questions :**

### TOPIC:CONSTRUCTOR IN JAVA

### 1 What is a [Constructor in Java](https://javainterviewpoint.com/constructor-in-java/)?

**Constructor** is just like a method in [**Java**](https://javainterviewpoint.com/category/core-java/) that is used to initialize the state of an object and will be invoked during the time of object creation.

### 2.What are the Rules for defining a constructor?

1. Constructor **name** should be the same as the class name
2. It **cannot** contain any **return type**
3. It **can** have all **Access Modifiers** are allowed (private , public, protected, default)
4. It **Cannot** have any **Non Access Modifiers** (final ,static, abstract, synchronized)
5. **No return** statement is allowed
6. It **can** take any number of **parameters**
7. Constructor can **throw exception**, we can have**throws clause**

### ****3. Can we have a Constructor in an Interface?****

**No**, We cannot have a Constructor defined in an **Interface**.

### ****4. What is a No-arg constructor?****

Constructor **without arguments** is called no-arg constructor. In Java Default constructor is a no-arg constructor.

class Demo

{

public Demo()

{

//No-arg constructor

}

}

### ****5. Can we have a class with no Constructor in it ? What will happen during object creation ?****

**Yes**, we can have a class with no constructor, When the compiler encounters a class with no constructor then it will automatically create a default constructor for you.

### 6. ****Can we have both Default Constructor and Parameterized Constructor in the same class?****

**Yes**, we have both Default Constructor and Parameterized Constructor in the same class.

### ****7.Can a Constructor return any value ?****

A Constructor cannot return any explicit value but implicitly it will be returning the instance of the class.

### ****8. Will compiler create the Default Constructor when we already have a Constructor defined in the class ?****

**No,** the compiler will not create the Default Constructor when we already have a Constructor defined.

### 9.Can an abstract class in Java have a constructor?

**Yes,** an abstract class can have a constructor.

### 10.Why constructors cannot be final in Java?

When you set a method as final, then” The method cannot be overridden by any class”, but **Constructor** by JLS ( [**Java Language Specification**](https://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.8.3) ) definition can’t be overridden. A constructor is not inherited, so there is no need for declaring it as **final**.

### 11.Why constructors cannot be abstract in Java?

When you set a method as abstract, then “The method doesn’t or cannot have body”. A constructor will be automatically called when object is created. It cannot lack a body moreover an abstract constructor could never be implemented.

### 12. Why constructors cannot be static in Java?

When you set a method as static, it means “The Method belong to class and not to any particular object” but a constructor is always invoked with respect to an object, so it makes no sense for a constructor to be **static**.

**TOPIC:INHERITANCE IN JAVA**

1. What is Inheritance?

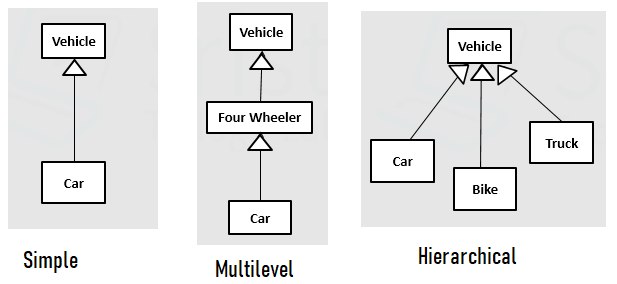
Inheritance is one important concept of OOPs. In Inheritance, one class can inherit the properties(instance variables) and behavior(methods) of another class. Few important terminologies that we use often are given below

**Super Class:**This is the class from which the properties(instance variables) and behavior(methods) are inherited. Also called as base class or parent class  
  
**Sub Class:**This is the class which inherits the properties and behavior from another class. Also called as derived class or child class. The sub class can access all the methods and variables except the private members. It also can have its own instance variables and methods.  
  
**extends:**Use the keyword extends to extend a class. The syntax is as follows  
**class Manager extends Employee**

##### **2.** Why do we need inheritance ?

Inheritance supports **code reusability**.  
  
Assume we have three classes Vehicle,Car,Bike with similar properties as brand,model,price and method as void getMileage(). Instead of creating the instance variables in both the classes, we can add the variables in just one class which is generic – in this case Vehicle class. Then make the Car and the Bike class inherit the properties and methods of Vehicle class.  
Remember while doing so, there should be an **IS-A** relationship between the classes.That is Car is a Vehicle, Bike is a Vehicle.  
  
Let me talk about a wrong scenario. Assume we have two classes as Employee and Student with similar properties as name,age,mobileno, city and method as void printDetails(). Is it right to apply inheritance here?. Try applying**IS-A** rule. Student is an Employee or Employee is a Student. Oh, that sounds weird. So don’t apply inheritance  
  
Remember the rule – Inheritance means **IS-A** relationship and is used for **code reusability**

##### 3. What type of inheritance does java support?

Java supports simple, multilevel and hierarchical inheritance.  
  
  
  
  
**Simple:**This is one level of inheritance. One class inherits from another class. In this case, class Car inherits from class Vehicle – **class Car extends Vehicle**  
  
**Multilevel:**This type of inheritance goes down for multiple levels –  
class Car inherits from FourWheeler, class FourWheeler inherits from class Vehicle  
class Vehicle is the sub class of Object class. So, automatically class Car can access the properties of Vehicle class.  
  
**Hierarchical:**Multiple classes can inherit from a single class or A class can have multiple subclasses. In this case, Bike,Car,Truck are sub classes of Vehicle.

##### 4. What is the super class for all the classes?

**Object** class from **java.lang package** is the super class for all the classes.

##### 5.What is the use of super keyword?

**‘super’** keyword should be always in the first line of any constructor. If it is not added by you, it will added automatically by the compiler. The purpose of super is to call and initialize the super class instance variables first. Always the super class instance variables are initialized first and then the subclass variables are initialized.  
Remember the parameters within **super()** should match atleast one constructor of the super class.

##### 6.Can a class extend more than one class?

No. Java does not support multiple inheritance. Please refer question-3. However, whatever classes you create will become a sub class of Object class

##### 7.Can I access the sub class methods using a super class object?

No. You cannot access the subclass methods/properties using the super class object.Only the sub class object can access the properties of the super class.(one real example – The son/daughter can access the fathers’ mobile. But the father cannot access the son/daughters’ mobile)

##### 8.What is the difference between Inheritance and Polymorphism?

Inheritance and Polymorphism complement each other. Runtime Polymorphism, which also called as Overriding happens with inheritance support. That is overriding happens only in a super class, subclass scenario.

### ****TOPIC:GARBAGE COLLECTION IN JAVA****

### ****What Is Garbage Collection and What Are Its Advantages?****

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.

An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused object, or unreferenced object, is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed.

The biggest advantage of garbage collection is that it removes the burden of manual memory allocation/deallocation from us so that we can focus on solving the problem at hand.

### ****Are There Any Disadvantages of Garbage Collection?****

Yes. Whenever the garbage collector runs, it has an effect on the application's performance. This is because all other threads in the application have to be stopped to allow the garbage collector thread to effectively do its work.

Depending on the requirements of the application, this can be a real problem that is unacceptable by the client. However, this problem can be greatly reduced or even eliminated through skillful optimization and garbage collector tuning and using different GC algorithms.

### ****What Happens When There Is Not Enough Heap Space to Accommodate Storage of New Objects?****

If there is no memory space for creating a new object in Heap, Java Virtual Machine throws OutOfMemoryError or more specifically**java.lang.OutOfMemoryError heap space.**

**Where are objects created in memory? On stack or heap?**

**Answer**.  Let’s start garbage collection interview question with very basic question. All Java objects are always created on **heap** in java.

**What is Garbage Collection process in java?**

**Answer**.  Basic garbage collection interview question. Definitely all developers must know it :)

GC (Garbage collection) is the process by which JVM cleans objects (unused objects) from heap to reclaim heap space in java.

**What is Automatic Garbage Collection in JVM heap memory in java?**

**Answer**.  Very important thing you must know in garbage collection interview.

**Automatic garbage collection** is the process of

* **Identifying objects which are in use** in java heap memory and
* **Which objects are not in use** in java heap memory and
* **deleting the unused objects** in java heap memory.

**How to Identify objects which are in use in JVM heap memory in java?**

**Answer**. It is very **basic** garbage collection interview question.

**Objects** in use (or **referenced objects**) are those objects which are still needed by java program, some part of java program is still pointing to that object.

**Which objects are not in use in JVM heap memory in java?**

**Answer**. Another very **basic** garbage collection interview question.

**Objects not** in use (or **unreferenced objects**) are those objects which are not needed by java program, no part of java program is pointing to that object.

So, these unused objects can be cleaned in GC (garbage collection) process and memory used by an unreferenced object can be reclaimed.

**TOPIC :DATA ABSTRACTION IN JAVA**

**1)Abstract class must have only abstract methods. True or false?**

False. Abstract methods can also have concrete methods.

**2) Is it compulsory for a class which is declared as abstract to have at least one abstract method?**

Not necessarily. Abstract class may or may not have abstract methods.

**3) Can we use “abstract” keyword with constructor, Instance Initialization Block and Static Initialization Block?**

No. Constructor, Static Initialization Block, Instance Initialization Block and variables can not be abstract.

**4) Why final and abstract can not be used at a time?**

Because, final and abstract are totally opposite in nature. A final class or method can not be modified further where as abstract class or method must be modified further. “final” keyword is used to denote that a class or method does not need further improvements. “abstract” keyword is used to denote that a class or method needs further improvements.

**5) Can we instantiate a class which does not have even a single abstract methods but declared as abstract?**

No, We can’t instantiate a class once it is declared as abstract even though it does not have abstract methods.

**6) Can we declare abstract methods as private? Justify your answer?**

No. Abstract methods can not be private. If abstract methods are allowed to be private, then they will not be inherited to sub class and will not get enhanced.

**7) We can’t instantiate an abstract class. Then why constructors are allowed in abstract class?**

It is because, we can’t create objects to abstract classes but we can create objects to their sub classes. From sub class constructor, there will be an implicit call to super class constructor. That’s why abstract classes should have constructors. Even if you don’t write constructor for your abstract class, compiler will keep default constructor.

**8) Can we declare abstract methods as static?**

No, abstract methods can not be static.

**9) Can a class contain an abstract class as a member?**

Yes, a class can have abstract class as it’s member.

**10) Abstract classes can be nested. True or false?**

True. Abstract classes can be nested i.e an abstract class can have another abstract class as it’s member.

**11) Can you create an instance of abstract class?**

No, you can not create instance of abstract class in Java, they are incomplete. Even though, if your abstract class don’t contain any abstract method, you can not create instance of it. By making a class abstract,  you told compiler that, it’s incomplete and should not be instantiated. Java compiler will throw error, when a code tries to instantiate abstract class.

**12) Is it necessary for an abstract class to have an abstract method?**

No, It’s not mandatory for an abstract class to have any abstract method. You can make a class abstract in Java, by just using abstract keyword in class declaration. Compiler will enforce all structural restriction, applied to abstract class, e.g. now allowing to create any instance. By the way, it’s debatable whether you should have abstract method inside abstract class or interface. In my opinion, abstract class should have abstract methods, because that’s the first thing programmer assumes, when he see that class. That would also go nicely along principle of least surprise.

**13) Difference between abstract class and interface in Java?**

This is the most important and one of the classic Java Interview question. I don’t know, how many times I have seen this question at all most all levels of Java interviews. One reason, which makes this question interesting is the ability to produce example. It’s easy to answers questions on core OOPS concepts like [Abstraction](http://javarevisited.blogspot.com/2010/10/abstraction-in-java.html), [Encapsulation](http://javarevisited.blogspot.com/2012/03/what-is-encapsulation-in-java-and-oops.html), [Polymorphism](http://javarevisited.blogspot.com/2011/08/what-is-polymorphism-in-java-example.html) and [Inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html), but when it comes to subtle points like this, candidate more often fumbled. You can see this post for all syntactical difference between abstract class and interface, but it deserve a post on it’s own.

**14) When do you favor abstract class over interface?**

This is the follow-up of previous interview questions on abstract class and interface. If you know syntactical difference, you can answer this question quite easily, as they are the one, which drives the decision. Since it’s almost impossible to add a new method on a published interface, it’s better to use abstract class, when evolution is concern. Abstract class in Java evolves better than interface. Similarly, if you have too many methods inside interface, you are creating pain for all it’s implementation, consider providing an abstract class for default implementation. This is the pattern followed in Java collection package, you can see AbstractList provides default implementation for List interface.

**15) What is abstract method in Java?**

An abstract method is a method without body. You just declare method, without defining it and use abstract keyword in method declaration.  All method declared inside [Java Interface](http://javarevisited.blogspot.com/2012/04/10-points-on-interface-in-java-with.html) are by default abstract. Here is an example of abstract method in Java

                public void abstract printVersion();

Now, In order to implement this method, you need to extend abstract class and [override](http://javarevisited.blogspot.com/2011/12/method-overloading-vs-method-overriding.html) this method.

16. **What is Abstraction in Java?**

Ans: [Abstraction in Java](https://www.scientecheasy.com/2020/05/java-abstraction.html/) is a technique by which we can hide the data that is not required to users. It hides all unwanted data so that users can work only with the required data.

**17) Can abstract class contains main method in Java ?**

Yes, abstract class can contain [main method](http://javarevisited.blogspot.sg/2011/12/main-public-static-java-void-method-why.html), it just another static method and you can execute Abstract class with main method, until you don’t create any instance.

18. **How to achieve or implement Abstraction in Java?**

Ans: There are two ways to implement abstraction in java. They are as follows:

a) Abstract class (0 to 100%)  
b) Interface (100%)

**19. What is Abstract class in Java? How to define it?**

Ans: An abstract class in java is a class that is declared with an abstract keyword.

20.**When to use Abstract method in Java?**

Ans: An abstract method can be used

a) When the same method has to perform different tasks depending on the object calling it.  
b) When you need to be overridden in its non-abstract subclasses.

**21.Is abstract class a pure abstraction in Java?**

Ans: No, It provides 0 to 100% abstraction.

**22. Is it possible to create an object of abstract class in Java?**

Ans: No. It is not possible but we can create an object of its subclass.

**23. Is it possible that an abstract class can have without any abstract method?**

Ans: Yes.

**24. Can an abstract class have constructor?**

Ans: Yes.

**25. Is abstract class allow to define private, final, static, and concrete methods?**

Ans: Yes.

**26. Is it possible to achieve multiple inheritance through abstract class?**

Ans: No.

**27. Can we make an abstract class without abstract keyword?**

Ans: No, a class must be declared with abstract keyword to make an abstract class.

**28. Can we define an abstract method inside non-abstract class (concrete class)?**

Ans: No, we cannot define an abstract method in non-abstract class.

29.**What is the advantage of Abstract class in Java?**

Ans: The main advantages of using abstract class are as follows:

* Abstract class makes programming better and more flexible by giving the scope of implementing abstract methods.
* Programmer can implement abstract method to perform different tasks depending on the need.
* We can easily manage code.

**TOPIC:FILE HANDLING IN JAVA**

|  |  |
| --- | --- |
| |  | | --- | | **What is a stream and what are the types of Streams and classes of the Streams?** | |
| Ans. A Stream is an abstraction that either produces or consumes information. There are two types of Streams :  Byte Streams: Provide a convenient means for handling input and output of bytes.  Character Streams: Provide a convenient means for handling input & output of characters.  Byte Streams classes: Are defined by using two abstract classes, namely InputStream and OutputStream.  Character Streams classes: Are defined by using two abstract classes, namely Reader and Writer. |

|  |  |
| --- | --- |
| |  | | --- | | **What is Scanner class used for ? when was it introduced in Java ?** | |
| Ans. Scanner class introduced in Java 1.5 for reading Data Stream from the imput device. Previously we used to write code to read a input using DataInputStream. After reading the stream , we can convert into respective data type using in.next() as String ,in.nextInt() as integer, in.nextDouble() as Double etc |

|  |  |
| --- | --- |
| |  | | --- | | **What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?** | |
| Ans. The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented |

**What is Java I/O ?  
A:**Java I/O (Input and Output) is used to process the input and produce the output. Java makes use of the stream concepts to make I/O operation fast. The java.io package contains all the classes required for input and output operations

**TOPIC:EXCEPTION HANDLING IN JAVA**

**1) What is an exception?**

Exception is an abnormal condition which occurs during the execution of a program and disrupts normal flow of the program. This exception must be handled properly. If it is not handled, program will be terminated abruptly.

**2) How the exceptions are handled in java? OR Explain exception handling mechanism in java?**

Exceptions in java are handled using try, catch and finally blocks.

try block : The code or set of statements which are to be monitored for exception are kept in this block.

catch block : This block catches the exceptions occurred in the try block.

finally block : This block is always executed whether exception is occurred in the try block or not and occurred exception is caught in the catch block or not.

**3) What is the difference between error and exception in java?**

Errors are mainly caused by the environment in which an application is running. For example, OutOfMemoryError happens when JVM runs out of memory. Where as exceptions are mainly caused by the application itself. For example, NullPointerException occurs when an application tries to access null object.

Click [here](https://javaconceptoftheday.com/difference-between-error-vs-exception-in-java/) to see more about Error Vs Exception in java.

**4) Can we keep other statements in between try, catch and finally blocks?**

No. We shouldn’t write any other statements in between try, catch and finally blocks. They form a one unit.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | try  {      // Statements to be monitored for exceptions  }    //You can't keep statements here    catch(Exception ex)  {      //Cathcing the exceptions here  }    //You can't keep statements here    finally  {      // This block is always executed  } |

**5) Can we write only try block without catch and finally blocks?**

No, It shows compilation error. The try block must be followed by either catch or finally block. You can remove either catch block or finally block but not both.

**6) There are three statements in a try block – statement1, statement2 and statement3. After that there is a catch block to catch the exceptions occurred in the try block. Assume that exception has occurred in statement2. Does statement3 get executed or not?**

No. Once a try block throws an exception, remaining statements will not be executed. control comes directly to catch block.

**7) What is unreachable catch block error?**

When you are keeping multiple catch blocks, the order of catch blocks must be from most specific to most general ones. i.e sub classes of Exception must come first and super classes later. If you keep super classes first and sub classes later, compiler will show unreachable catch block error.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class ExceptionHandling  {      public static void main(String[] args)      {          try          {              int i = Integer.parseInt("abc");   //This statement throws NumberFormatException          }            catch(Exception ex)          {              System.out.println("This block handles all exception types");          }            catch(NumberFormatException ex)          {              //Compile time error              //This block becomes unreachable as              //exception is already caught by above catch block          }      }  } |

**8) Explain the hierarchy of exceptions in java?**

Click [here](https://javaconceptoftheday.com/hierarchy-exceptions-java/) to see the hierarchy of exceptions in java.

**9) What are run time exceptions in java. Give example?**

The exceptions which occur at run time are called as run time exceptions. These exceptions are unknown to compiler. All sub classes of java.lang.RunTimeException and java.lang.Error are run time exceptions. These exceptions are unchecked type of exceptions. For example, NumberFormatException, NullPointerException, ClassCastException, ArrayIndexOutOfBoundException, StackOverflowError etc.

**10) What is OutOfMemoryError in java?**

OutOfMemoryError is the sub class of java.lang.Error which occurs when JVM runs out of memory.

**11) what are checked and unchecked exceptions in java?**

Checked exceptions are the exceptions which are known to compiler. These exceptions are checked at compile time only. Hence the name checked exceptions. These exceptions are also called compile time exceptions. Because, these exceptions will be known during compile time.

Unchecked exceptions are those exceptions which are not at all known to compiler. These exceptions occur only at run time. These exceptions are also called as run time exceptions. All sub classes of java.lang.RunTimeException and java.lang.Error are unchecked exceptions.

Click [here](https://javaconceptoftheday.com/checked-unchecked-exceptions-java/) to see more about checked and unchecked exceptions.

**12) What is the difference between ClassNotFoundException and NoClassDefFoundError in java?**

Click [here](https://javaconceptoftheday.com/classnotfoundexception-vs-noclassdeffounderror-in-java/) to see the differences between ClassNotFoundException and NoClassDefFoundError in java.

**13) Can we keep the statements after finally block If the control is returning from the finally block itself?**

No, it gives unreachable code error. Because, control is returning from the finally block itself. Compiler will not see the statements after it. That’s why it shows unreachable code error.

**14) Does finally block get executed If either try or catch blocks are returning the control?**

Yes, finally block will be always executed no matter whether try or catch blocks are returning the control or not.

**15) Can we throw an exception manually? If yes, how?**

Yes, we can throw an exception manually using throw keyword.

**16) What is the use of throws keyword in java?**

**17) Why it is always recommended that clean up operations like closing the DB resources to keep inside a finally block?**

Because finally block is always executed whether exceptions are raised in the try block or not and raised exceptions are caught in the catch block or not. By keeping the clean up operations in finally block, you will ensure that those operations will be always executed irrespective of whether exception is occurred or not.

## 18. What Is an Exception in Java?

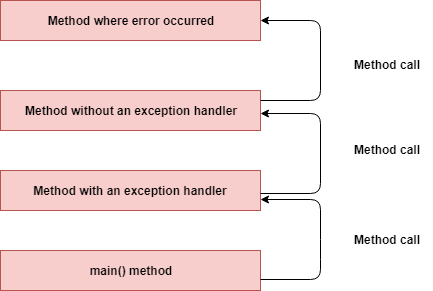
An exception is an event that occurs during the execution of a program and disrupts the normal flow of the program's instructions.

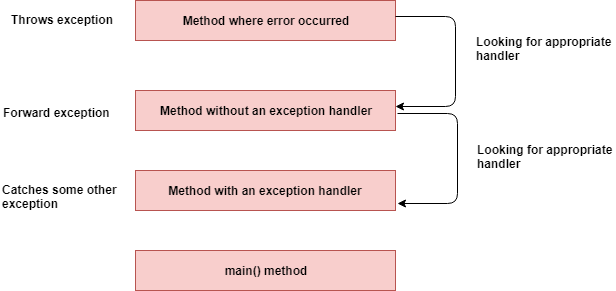
Read more about exceptions in Java in the [Java Exception Handling Guide](http://www.javaguides.net/2018/08/java-exception-handling-guide.html)!

## 19. How Does Exception Handling Work in Java?

Below steps demonstrates how the exception handling works in Java:

**Step 1:** When an error occurs within a method, the method creates an object and hands it off to the runtime system this object is called an exception object. The exception object contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

**Step 2:** After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of methods is known as the call stack. The following diagram shows the call stack of three method calls, where the first method called has the exception handler.  
  
[](https://4.bp.blogspot.com/-0Wo56QV6xn4/XCS7VWHL0qI/AAAAAAAAFNo/8O8OQJlQmIkqoMITTOt8T-H37z5flfeTgCLcBGAs/s1600/exception-call+stack.png)

**Step 3:** The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler. The search begins with the method in which the error occurred and proceeds through the call stack in the reverse order in which the methods were called. When an appropriate handler is found, the runtime system passes the exception to the handler.   
An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.  
  
**Step 4:**The exception handler chosen is said to catch the exception. If the runtime system exhaustively searches all the methods on the call stack without finding an appropriate exception handler, as shown in the following diagram, the runtime system (and, consequently, the program) terminates.  
[](https://3.bp.blogspot.com/-FNi4uc7THNY/XCS7c7FvTpI/AAAAAAAAFNs/3YB9a811_yEss_ky-ku675sg6OH2HJdHwCLcBGAs/s1600/exception-searching-call+stack.png)

## 20. What Are  Exception Handling Keywords in Java?

Java exception handling is managed via five keywords:

**1. try:**Enclose the code that might throw an exception within a try block. If an exception occurs within the try block, that exception is handled by an exception handler associated with it. The try block contains at least one catch block or finally block.

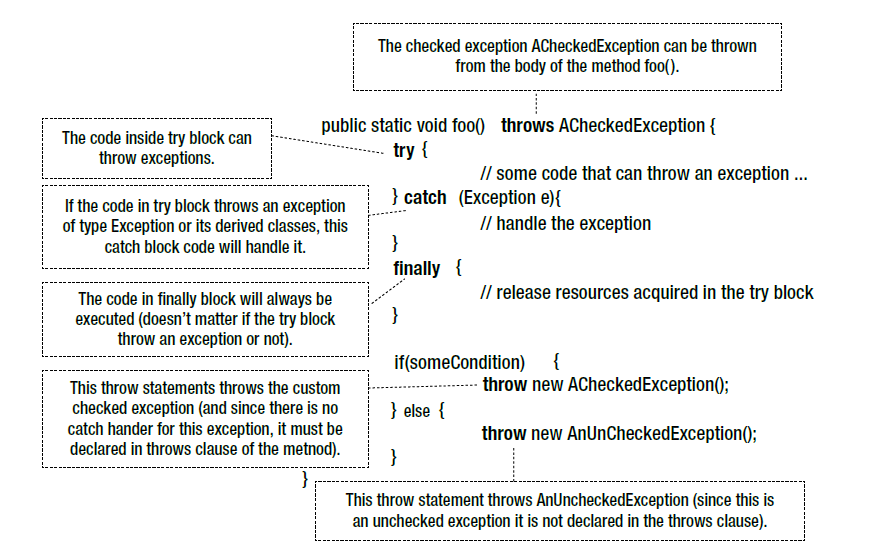
**2. catch:** The Java catch block is used to handle the exception. It must be used after the try block only. You can use multiple catch blocks with a singletry.

**3. throw:** Sometimes, we explicitly want to create an exception object and then throw it to halt the normal processing of the program. The throw keyword is used to throw an exception to the run-time to handle it.

**4. throws:** When we are throwing any checked exception in a method and not handling it, then we need to use the throws keyword in the method signature to let the caller program know that the exceptions might be thrown by the method. The caller method might handle these exceptions or propagate it to its caller method using the throws keyword. We can provide multiple exceptions in the throws clause and it can be used with the main()method also.

**5. finally:**The finally block is optional and can be used only with thetry-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use finally block. The finally block gets executed always, whether an exception occurs or not.

This diagram provides a summary of the usage of these keywords.

[](https://1.bp.blogspot.com/-VBriy2lMQrs/W8IlIHxwAMI/AAAAAAAAERU/u0OlZUp30VkB_FV44BE0n3R9LcMkEFTWQCLcBGAs/s1600/exception-handling-keywords.PNG)

## 21. What Is the Purpose of the Throw and Throws keywords?

Thethrows keyword is used to specify that a method may raise an exception during its execution. It enforces explicit exception handling when calling a method:

1

public void simpleMethod() throws Exception {

2

// ...

3

}

Thethrow keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

1

if (task.isTooComplicated()) {

2

throw new TooComplicatedException("The task is too complicated");

3

}

## 22. How Can You Handle an Exception?

You can handle an exception by using a try-catch-finallystatement:

1

try {

2

// ...

3

} catch (ExceptionType1 ex) {

4

// ...

5

} catch (ExceptionType2 ex) {

6

// ...

7

} finally {

8

// ...

9

}

The block of code in which an exception may occur is enclosed in a try block. This block is also called “protected” or “guarded” code. If an exception occurs, the catch block that matches the exception being thrown is executed. If not, all catch blocks are ignored. The finally block is always executed after the try block exits, whether an exception was thrown inside it or not.

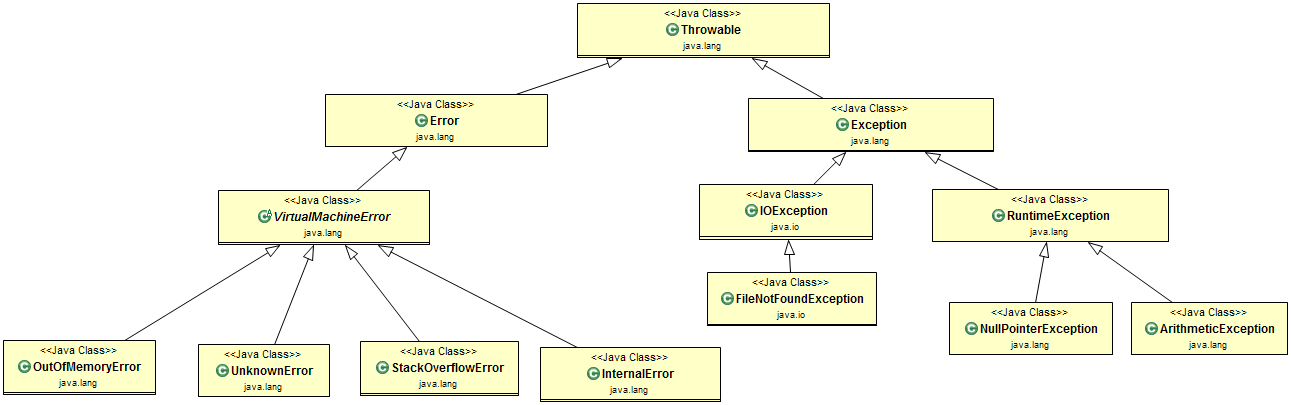
## 22. Explain the Java Exception Hierarchy.

The objects that inherit from the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank) class that includes direct descendants (objects that inherit directly from the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank) class) and indirect descendants (objects that inherit from children or grandchildren of the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank) class).

[Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) has two direct descendants:

1. Error Class
2. Exception Class

The figure below illustrates the class hierarchy of the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank) class and its most significant subclasses.

[](https://2.bp.blogspot.com/--f8P24mVPK0/W3v8Qq2JzrI/AAAAAAAADSA/PR-7hjBxP1QPhzc5UsYyZBgFseJ7QM2XgCLcBGAs/s1600/exception-handling.png)

**Error Class:**When a dynamic linking failure or other hard failures in the Java Virtual Machine occurs, the virtual machine throws an Error.

Examples: VirtualMachineError,  OutOfMemoryError, UnKnownError,  StackOverflowError, etc.

**Exception Class:**Most programs throw and catch objects that derive from the exception class. An exception indicates that a problem occurred, but it is not a serious system problem. For example, when dealing with the FileNotFoundException, we should catch this exception and provide a useful message to the user and log it properly for debugging purposes. The exception is the parent class of all checked exceptions.

**RuntimeException Class:**This provides one exception subclass, RuntimeException, that is reserved for exceptions that indicate an incorrect use of an API. An example of a runtime exception is the NullPointerException, which occurs when a method tries to access a member of an object through a null reference.

## 23. How Can you Catch Multiple Exceptions?

There are three ways to handle multiple exceptions in a block of code.

The first is to use a catch block that can handle all exception types being thrown:

1

try {

2

// ...

3

} catch (Exception ex) {

4

// ...

5

}

You should keep in mind that the recommended practice is to use exception handlers that are as accurate as possible.

Exception handlers that are too broad can make your code more error-prone, catch exceptions that weren’t anticipated, and cause unexpected behavior in your program.

The second way is implementing multiple catch blocks:

1

try {

2

// ...

3

} catch (FileNotFoundException ex) {

4

// ...

5

} catch (EOFException ex) {

6

// ...

7

}

Note that if the exceptions have an inheritance relationship, the child type must come first and the parent type later. If we fail to do this, it will result in a compilation error.

The third is to use a multi-catch block:

1

try {

2

// ...

3

} catch (FileNotFoundException | EOFException ex) {

4

// ...

5

}

## 24. What Is the Difference Between Checked and Unchecked Exceptions in Java?

1. Checked exceptions should be handled in the code using a try-catch block, or else, the method should use the throws keyword to let the caller know about the checked exceptions that might be thrown from the method. Unchecked Exceptions are not required to be handled in the program or to mention them in thethrows clause of the method.

2. The exception is the superclass of all checked exceptions, whereas RuntimeException is the superclass of all unchecked exceptions. Note that RuntimeException is the child class ofException.

3. Checked exceptions are error scenarios that require being handled in the code, or else, you will get a compile-time error. For example, if you use FileReader to read a file, it throws the FileNotFoundException and we must catch it in the [try-catch block](http://www.javaguides.net/2018/08/java-trycatch-block.html) or throw it again to the caller method. Unchecked exceptions are mostly caused by poor programming, for example, the NullPointerException when invoking a method on an object reference without making sure that it’s not null. I can write a method to remove all the vowels from the string. It’s the caller's responsibility to make sure not to pass a null string. I might change the method to handle these scenarios, but ideally, the caller should take care of this.

4. Checked and unchecked exceptions are also known as compile-time and run-time exceptions respectively.

## 25. What Is the Difference Between Throw and Throws Keywords in Java?

The throws keyword is used with a method signature to declare exceptions that the method might throw, whereas the throw keyword is used to disrupt the flow of a program and handing over the exception object to run-time to handle it.

## 26. What Is the Difference Between an Exception and an Error?

An exception is an event that represents a condition from which it is possible to recover, whereas an error represents an external situation usually impossible to recover from.

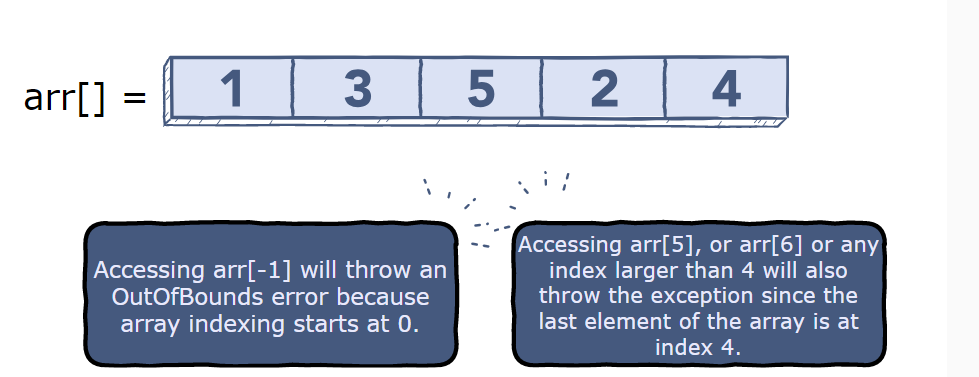
All errors thrown by the JVM are instances of Error or one of its subclasses. The more common ones include:

* OutOfMemoryError – thrown when the JVM cannot allocate more objects because it is out memory and the garbage collector was unable to make more available.
* StackOverflowError – occurs when the stack space for a thread has run out. This is typically because an application recurses too deeply.
* ExceptionInInitializerError – signals that an unexpected exception occurred during the evaluation of a static initializer.
* NoClassDefFoundError – is thrown when the classloader tries to load the definition of a class and couldn’t find it, usually because the required class files were not found in the classpath.
* UnsupportedClassVersionError – occurs when the JVM attempts to read a class file and determines that the version in the file is not supported, normally because the file was generated with a newer version of Java

Although an error can be handled with a try statement, this is not a recommended practice since there is no guarantee that the program will be able to do anything reliably after the error was thrown.

## 27. What Is the OutOfMemoryError in Java?

The **ArrayIndexOutOfBounds exception** is thrown if a program tries to access an array index that is negative, greater than, or equal to the length of the array.



The ArrayIndexOutOfBounds exception is a run-time exception. Java’s compiler does not check for this error during compilation.

1