Java Mock Session Questions:

What is a Class ?

What is a Constructor, whats the use of it?

7.what are oops principles ?

Encapsulation

data abstraction

polymorphism

inheritance

What is Encalsulation ?

Wrapping states and behaviours...by implementing the class we can achieve encapsulation.

We do this getters and setters private variables can be accesed other classes using

9.what is polymorphism? types of polymorphism ?

implementing mutiple functinalities with the same name : two types

1.static polymorphism

2.dynamic polymorphism

10.what are static and dynamic polymorphisms.

static : method which bind at complie time and which will execute at runtime or early binding time binding , compile time polymorphisms.

1.What is Encapsulation?

2.Explain This keyword ?

**this** keyword in java – this is used to refer the member(instance variables,methods) of current object from method or constructor.

It can be used inside a method or a constructor of the class.

this is a reference to current object of execution.

1. Diff between Method Overloading and Method Overriding ?
2. What is Constructor in a class , Can an abstract class have a constructor?

### ANS: Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation. Rules for creating java constructor

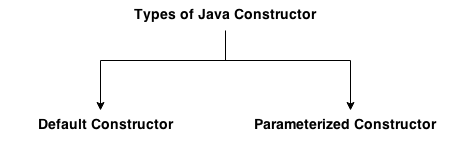
There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name
2. Constructor must have no explicit return type

Types of java constructors

There are two types of constructors:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

* 

#### If there is no constructor in a class, compiler automatically creates a default constructor.

Default constructor provides the default values to the object like 0, null etc. depending on the type.

1. **class** Student3{
2. **int** id;
3. String name;
5. **void** display(){System.out.println(id+" "+name);}
7. **public** **static** **void** main(String args[]){
8. Student3 s1=**new** Student3();
9. Student3 s2=**new** Student3();
10. s1.display();
11. s2.display();
12. }
13. }
14. Output:
15. 0 null
16. 0 null
17. **Explanation:**In the above class,you are not creating any constructor so compiler provides you a default constructor.Here 0 and null values are provided by default constructor.

You would define a constructor in an abstract class if you are in one of these situations:

* you want to perform some initialization (to fields of the abstract class) before the instantiation of a subclass actually takes place
* you have defined final fields in the abstract class but you did not initialize them in the declaration itself; in this case, you MUST have a constructor to initialize these fields

Note that:

* you may define more than one constructor (with different arguments)
* you can (should?) define all your constructors protected (making them public is pointless anyway)
* your subclass constructor(s) can call one constructor of the abstract class; it may even **have to** call it (if there is no no-arg constructor in the abstract class)

In any case, don't forget that if you don't define a constructor, then the compiler will automatically generate one for you (this one is public, has no argument, and does nothing).

1. Abstract and Interface ?
2. Explain the java Keywords This, Super and Static ?
3. **In how many ways you can create string objects in java?**

There are two ways to create string objects in java. One is using *new* operator and another one is using string *literals*. The objects created using new operator are stored in the heap memory and objects created using string literals are stored in string constant pool.

|  |  |
| --- | --- |
| 1  2 | String s1 = new String("abc");          //Creating string object using new operator    String s2 = "abc";        //Creating string object using string literal |

1. Why String is Immutable in Java ? Because java uses the concept of string literal. Suppose there are 5 reference variables,all referes to one object "sachin".If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

 How many objects will be created in the following code?

1. String s1="Welcome";
2. String s2="Welcome";
3. String s3="Welcome";

Only one object.

1. **Which one will you prefer among “==” and equals() method to compare two string objects?**
2. I prefer *equals()* method because it compares two string objects based on their content. That provides more logical comparison of two string objects. If you use *“==”* operator, it checks only references of two objects are equal or not. It may not be suitable in all situations. So, rather stick to *equals()* method to compare two string objects. [[more](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)]
3. **What is string constant pool?**

String objects are most used data objects in Java. Hence, java has a special arrangement to store the string objects. String Constant Pool is one such arrangement. String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content.

Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the string constant pool with the same content, then it returns the reference of that object. It doesn’t create a new object. If the content is different from the existing objects then only it creates new object. **Where exactly string constant pool is located in the memory?**Inside the heap memory. JVM reserves some part of the heap memory to store string objects created using string literals.

1. String buffer(thread safe) and builder ? mutable ?
2. Program to reverse string ? check for panlindrome ?
3. Auto boxing ? wrapper classes ?

**Shen Watsapp Questions :**

1. **CAN WE HAVE MULTIPLE MAIN METHODS IN CLASS ?**

Yes, Java Program can have multiple main() methods as Java supports overloading of main method also. Overloading of themethods differentiate by the number of parameters and the type of the arguments passed into themethod.

Public static void main(String[] args) is the only main method with single String[] or String as parameter will be considered as the Entry point for the program.

We can have more classes that contain the main() in a program, and can chose to execute one class or another class.

But we can’t have more than one main method “main(String[] args)” within the same class.

'main' means public static void main(String[] args) which is entry point in java programs . BUT you can overload main methods that is you can change number of parameters and type of parameters which will be considered as a different function by the compiler ,that is   
public static void main(int i)  
{  
//code  
}  
Thus this method will not show any error because parameter is not of String type.

**Difference between HashMap and HashTable / HashMap vs HashTable**

HashMap and Hashtable both are used to store data in key and value form. Both are using hashing technique to store unique keys.

Difference between hashmap and hashtable  includes five point  namely Synchronization,Null keys and values,Iterating values , Fail fast iterator ,Performance,Superclass .  
    
**1. Synchronization or Thread Safe :**  This is the most important difference between two . HashMap is non synchronized and not thread safe.On the other hand, HashTable is thread safe and synchronized.  
When to use HashMap ?  answer is if your application do not require any multi-threading task, in other words hashmap is better for non-threading applications. HashTable should be used in multithreading applications.   
  
**2. Null keys and null values :**  Hashmap allows one null key and any number of null values, while Hashtable do not allow null keys and null values in the HashTable object.

**3. Iterating the values:**  Hashmap object values are iterated by using iterator .HashTable is the only class other than vector which uses enumerator to iterate the values of HashTable object.  
  
**4. Fail-fast iterator**  : The iterator in Hashmap is fail-fast iterator while the enumerator for Hashtable is not.According to [Oracle Docs](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html),  if the Hashtable is structurally modified at any time after the iterator is created in any way except the iterator's own remove method , then the iterator will throw ConcurrentModification Exception.  
Structural modification means adding or removing elements from the Collection object (here hashmap or hashtable) . Thus the enumerations returned by the Hashtable keys and elements methods are not fail fast. We have already explained the[difference between iterator and enumeration](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html).  
  
**5. Performance :**  Hashmap is much faster and uses less memory than Hashtable as former is unsynchronized . Unsynchronized objects are often much better in performance in compare to synchronized  object like Hashtable in single threaded environment.  
  
**6. Superclass and Legacy :**  Hashtable is a subclass of Dictionary class which is now obsolete in Jdk 1.7 ,so ,it is not used anymore. It is better off externally synchronizing a HashMap or using a ConcurrentMap implementation (e.g ConcurrentHashMap).HashMap is the subclass of the AbstractMap class. Although Hashtable and HashMap has different superclasses but they both are implementations of the *"Map"*  abstract data type.

### 3Q. What is difference between throw and throws?

|  |  |
| --- | --- |
| **throw keyword** | **throws keyword** |
| 1)throw is used to explicitly throw an exception. | throws is used to declare an exception. |
| 2)checked exceptions can not be propagated with throw only. | checked exception can be propagated with throws. |
| 3)throw is followed by an instance. | throws is followed by class. |
| 4)throw is used within the method. | throws is used with the method signature. |
| 5)You cannot throw multiple exception | You can declare multiple exception e.g. public void method()throws IOException,SQLException. |

# NullPointerException in Java

NullPointerException is a **RuntimeException** . Runtime exceptions are critical and cannot be caught at compile time. They crash the program at run time if they are not handled properly. When a class is instantiated, its object is stored in computer memory. The**NullPointerExceptions** occur when you try to use a reference that points to no location in memory (null) as though it were referencing an object. These include:

* Calling the instance method of a null object.
* Accessing or modifying the field of a null object.
* Throwing null as if it were a Throwable value.

The **NullPointerException** occurs when you declare a variable but did not create an object. If you attempt to dereference str before creating the object you get a NullPointerException.

Ex. String str=null;

System.out.prinln(str.length());

The first line declares a variable named str and set to null. In the second line, you try to find the length of the String without instantiate (or create) an object of type String using **"new"**keyword. When you compile the code, the compiler generate .class file without any errors. But, when you run the above code, you will get **"Exception in thread "main" java.lang.NullPointerException"** . Here this exception is raised because when you try to access members from a class using object reference that is initialized to null value.

In order to fix this **NullPointerException** you should always initialize your objects before you try to do anything with them.

5Q. static methods ca you over ride or not? Overriding depends on having an instance of a class. The point of polymorphism is that you can subclass a class and the objects implementing those subclasses will have different behaviors for the same methods defined in the superclass (and overridden in the subclasses). A static method is not associated with any instance of a class so the concept is not applicable.

**Exceptions , Types and how to handle ?**

An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

* A user has entered an invalid data.
* A file that needs to be opened cannot be found.
* A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

* **Checked exceptions** − A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

For example, if you use **FileReader** class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a *FileNotFoundException* occurs, and the compiler prompts the programmer to handle the exception.

### Example

[Live Demo](http://tpcg.io/9u4a5O)

import java.io.File;

import java.io.FileReader;

public class FilenotFound\_Demo {

public static void main(String args[]) {

File file = new File("E://file.txt");

FileReader fr = new FileReader(file);

}

}

If you try to compile the above program, you will get the following exceptions.

### Output

C:\>javac FilenotFound\_Demo.java

FilenotFound\_Demo.java:8: error: unreported exception FileNotFoundException; must be caught or declared to be thrown

FileReader fr = new FileReader(file);

^

1 error

**Note** − Since the methods **read()** and **close()** of FileReader class throws IOException, you can observe that the compiler notifies to handle IOException, along with FileNotFoundException.

* **Unchecked exceptions** − An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an *ArrayIndexOutOfBoundsExceptionexception* occurs.

### Example

[Live Demo](http://tpcg.io/7CUnsL)

public class Unchecked\_Demo {

public static void main(String args[]) {

int num[] = {1, 2, 3, 4};

System.out.println(num[5]);

}

}

If you compile and execute the above program, you will get the following exception.

### Output

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5

at Exceptions.Unchecked\_Demo.main(Unchecked\_Demo.java:8)

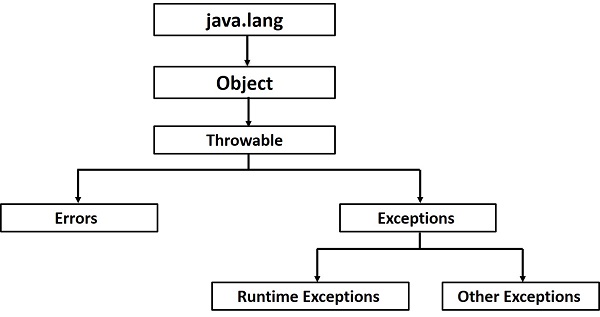
* **Errors** − These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

## Exception Hierarchy

All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.

Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example: JVM is out of memory. Normally, programs cannot recover from errors.

The Exception class has two main subclasses: IOException class and RuntimeException Class.



Try and Ctach blocks

WAIT AND NOTIFY:-

|  |
| --- |
| [**void notify()**](https://www.tutorialspoint.com/java/lang/object_notify.htm)  This method wakes up a single thread that is waiting on this object's monitor. |
| 7 | [**void notifyAll()**](https://www.tutorialspoint.com/java/lang/object_notifyall.htm)  This method wakes up all threads that are waiting on this object's monitor. |
| [**void wait()**](https://www.tutorialspoint.com/java/lang/object_wait.htm)  This method causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object. |
| 10 | [**void wait(long timeout)**](https://www.tutorialspoint.com/java/lang/object_wait_timeout.htm)  This method causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed. |
| 11 | [**void wait(long timeout, int nanos)**](https://www.tutorialspoint.com/java/lang/object_wait_nanos.htm)  This method causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. |

The Object class in Java[S](http://java.sun.com/)[W](http://en.wikipedia.org/wiki/Java_(programming_language)) has three final methods that allow threads to communicate about the locked status of a resource. These methods are wait(), notify(), and notifyAll(). A thread obtains a lock for a particular resource via a synchronized block with an instance of that resource. Suppose that a thread requires that another thread perform a certain action on the resource before it acts on the resource. That thread can synchronize on the resource and call the wait() method on resource. This says that the thread will wait until it has been notified that it can proceed to act. The wait() method can take an optional timeout value as a parameter. If this value is used, it means that the thread will either wait until it's notified or it will continue to execute once the timeout value has passed.

If a thread is required to perform a task on a resource before another thread operates on the resource (and the other thread is waiting via the wait() method on the resource), the thread needs synchronize on the resource. It can perform its actions on the resource. In order to notify the waiting thread once these actions have completed, the notify() method on the resource is called. This notifies the waiting thread that it can proceed to act. If multiple threads are waiting for the resource, there is no guarantee as to which thread will be given access to the resource. If it is desired for all waiting threads to be awoken, the notifyAll() method can be called on the resource.