JAVA INTERVIEW QUESTIONS CHAPTER WISE

Index

Chapter No	Name	No Of Questions
1	Introduction to Java architecture	4
2	Java Syntax and Code review	20
3	Encapsulation and polymorphism	30
4	Java Class Design	3
5	Advance Class Design	20
6	Inheritance and Java Interface	4
7	Collection and Generics	21
8	Strings	14
9	Exception And Assertions	9
10	Basic IO	11
11	NIO	2
12	Threads	6
13	Concurrency	8
14	JDBC	24
15	Localization	6

Sr No	Chapter 1	Introduction to Java and its Architecture
1	Why java is called as Platform Independent.	Platform independence means that we can write and compile the java code in one platform (eg Windows) and can execute the class in any other supported platform eg (Linux,Solaris,etc).
2	What is JVM	JVM stands for Java Virtual Machine. It's an abstract computer or virtual computer which runs the compiled java programs. Actually JVM is a software implementation which stands on the top of the real hardware platform and operating system. It provides abstraction between the compiled java program and the hardware and operating system.
3	What is JIT (Just-in-Time) Compilation?	Machine understands only binary language. So finally source code has to be compiled in binary format. Compiled and interpreter varies from the way they generate the binary files. When JVM compiles the class file he does not compile the full class file in one shot. Compilation is done on function basis or file basis. Advantage gained from this is that heavy parsing of original source code is avoided. Depending on need basis the compilation is done. This type of compilation is termed as JIT or Just-in-Time compile.
4	What is the difference between a JDK and a JVM?	JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

Sr No	Chapter 2	Java Syntax and Class Review
1	Name the primitive data types in java	Integers are byte,short,int,long Floating points are float,double Character char True/False Boolean.
2	Define class?	A class is a blue print from which individual objects are created. A class can contain fields and methods to describe the behavior of an object.
3	What kind of variables a class can consist of?	A class consists of Local variable, instance variables and class variables.
4	What is a Local Variable	Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and it will be destroyed when the method has completed.
5	What is a Instance Variable	Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded.
6	What is a Class Variable	These are variables declared with in a class, outside any method, with the static keyword.
7	What is Singleton class?	Singleton class control object creation, limiting the number to one but allowing the flexibility to create more objects if the situation changes.
8	What's the difference between a primitive type and a class type in Java?	Every variable in Java has a type, which essentially tells Java how that variable should be treated, and how much memory should be allocated for that variable. Java has basic types for characters, different kinds of integers, and different kinds of floating point numbers (numbers with a decimal point), and also types for the values true and false – char, int, float, and bool. All of these basic types are known as primitive types
9	What is a class type in Java?	Java classes, as you probably already know, are created to solve object oriented problems. Any object of a class has the type "class". Because an object of a class is more complex than a simple integer, boolean, or other "primitive" type, a variable naming an object is known to be a class type.
10	Class types vs Object types vs Reference types ?	You might be confused by all the different terminology used. So just to clarify, class types, object types, and reference types all mean the exact same thing – an object of a class. In this discussion, we will just be using the term class types to refer to objects of a class.
11	The differences between class and primitive types in Java ?	A variable of a class type – like a String – stores objects of its class differently from how variables of primitive types – like int or char – store their values. Every variable, whether it's of a primitive type or of a class type, is implemented as a location in computer memory. For a variable of a primitive type, the value of the variable is stored in the memory location assigned to the variable. So, if an integer variable is declared as "int x

		= 3”, then when we look at the memory location of “x”, there will be a “3” stored there just as expected. However, a variable of a class type only stores the memory address of where the object is located – not the values inside the object. So, if we have a class called “Some Class”, when we create an object like this: “Some Class an Object”, then when we look at “an Object” in memory, we will see that it does not store any of the variables that belong to that object in memory. Instead, the variable “an Object” just stores an address of another place in memory where all the details of “an Object” reside. This means that the object named by the variable is stored in some other location in memory and the variable contains only the memory address of where the object is stored. This memory address is called a reference to the object. If this is confusing, you will see an example of this further down.
12	Different memory requirements for variables of class types and primitive types	A value of a primitive type – like a type int – will always require the same amount of memory to store one value. There is a maximum value of type int – which means that values of type int will have a limit on their size. But – the big difference between a primitive type and a class type is that an object of a class type, like an object of the class String, can be of any size. The memory location for a variable of type String (a class type) is of a fixed size, so it cannot store a gigantic string like one that is 3,000 characters long. But, it can and does store the address of any string since there is a limit to the size of an address. So, remember that a variable of a class type stores an address of the object, and not the object values themselves – which are stored at that address in memory.
13	What is the difference between the Boolean & operator and the && operator?	If an expression involving the Boolean & operator is evaluated, both operands are evaluated. Then the & operator is applied to the operand. When an expression involving the && operator is evaluated, the first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The && operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.
14	What would you use to compare two String variables - the operator == or the method equals()?	I'd use the method equals() to compare the values of the Strings and the == to check if two variables point at the same instance of a String object.
15	What is the difference between the >> and >>> operators?	The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out
16	What is the difference between a while statement and a do statement?	while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute the body of a loop at least once.
17	Explain Garbage collection mechanism in Java?	Garbage collection is one of the most important features of Java. The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used. Garbage collection is also called automatic memory management as JVM automatically removes the unused variables/objects (value is null) from the memory. Every class inherits finalize() method from java.lang.Object, the finalize() method is called by garbage collector when it determines no more references to the object exists. In Java, it is good idea to explicitly assign null into a variable when no more in use. In Java on calling System.gc() and Runtime.gc(), JVM tries to recycle the unused objects, but there is no guarantee when all the objects will garbage collected. Garbage collection is an automatic process and can’t be forced. There is no guarantee that Garbage collection will start immediately upon request of System.gc().
18	Can an object’s finalize() method be invoked while it is reachable?	An object’s finalize() method cannot be invoked by the garbage collector while the object is still reachable. However, an object’s finalize() method may be invoked by other objects.

19	Does garbage collection guarantee that a program will not run out of memory?	Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to garbage collection.
20	What is the purpose of finalization?	The purpose of finalization is to give an unreachable object the opportunity to perform any cleanup, before the object gets garbage collected. For example, closing an opened database Connection.

Sr No	**Chapter 3**	**Encapsulation and polymorphism**
1	What are different types of access modifiers?	Access specifiers are keywords that determine the type of access to the member of a class. These keywords are for allowing privileges to parts of a program such as functions and variables. These are: • *Public* : accessible to all classes • *Protected* : accessible to the classes within the same package and any subclasses. • *Private* : accessible only to the class to which they belong • *Default :* accessible to the class to which they belong and to subclasses within the same package
2	What are the principle concepts of OOPS?	There are four principle concepts upon which object oriented design and programming rest. They are: Abstraction Polymorphism Inheritance Encapsulation (i.e. easily remembered as A-PIE).
3	What's the relation between Classes and Objects?	They look very much same but are not same. Class is a definition, while object is instance of the class created. Class is a blue print while objects are actual objects existing in real world. Example we have class CAR which has attributes and methods like Speed, Brakes, Type of Car etc. Class CAR is just a prototype, now we can create real time objects which can be used to provide functionality. Example we can create a Maruti car object with 100 km speed and urgent brakes.
4	What is Abstraction?	Abstraction refers to the act of representing essential features without including the background details or explanations.
5	What is Encapsulation?	Encapsulation is a technique used for hiding the properties and behaviours of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.
6		
7	What is the difference between abstraction and encapsulation?	Abstraction focuses on the outside view of an object (i.e. the interface) Encapsulation (information hiding) prevents clients from seeing it's inside view, where the behaviour of the abstraction is implemented. Abstraction solves the problem in the design side while Encapsulation is the Implementation. Encapsulation is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.
8	What is Inheritance?	Inheritance is the process by which objects of one class acquire the properties of objects of another class. A class that is inherited is called a superclass. The class that does the inheriting is called a subclass. Inheritance is done by using the keyword extends. The two most common reasons to use inheritance are: To promote code reuse To use polymorphism
9	What is	

<table>
<tr><td></td><td>Polymorphism?</td><td>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.</td></tr>
<tr><td><b>10</b></td><td>How does Java implement polymorphism?</td><td>(Inheritance, Overloading and Overriding are used to achieve Polymorphism in java).
Polymorphism manifests itself in Java in the form of multiple methods having the same name.
In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods).
In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).</td></tr>
<tr><td><b>11</b></td><td>Explain the different forms of Polymorphism.</td><td>There are two types of polymorphism one is Compile time polymorphism and the other is run time polymorphism. Compile time polymorphism is method overloading. Runtime time polymorphism is done using inheritance and interface.
Note: From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:
Method overloading
Method overriding through inheritance
Method overriding through the Java interface</td></tr>
<tr><td><b>12</b></td><td>What is runtime polymorphism or dynamic method dispatch?</td><td>In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.</td></tr>
<tr><td><b>13</b></td><td>What is Dynamic Binding?</td><td>Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.</td></tr>
<tr><td><b>14</b></td><td>What is Constructor?</td><td>A constructor is a special method whose task is to initialize the object of its class.
It is special because its name is the same as the class name.
They do not have return types, not even void and therefore they cannot return values.
They cannot be inherited, though a derived class can call the base class constructor.
Constructor is invoked whenever an object of its associated class is created.</td></tr>
<tr><td><b>15</b></td><td>How does the Java default constructor be provided?</td><td>If a class defined by the code does not have any constructor, compiler will automatically provide one no-parameter-constructor (default-constructor) for the class in the byte code. The access modifier (public/private/etc.) of the default constructor is the same as the class itself.</td></tr>
<tr><td><b>16</b></td><td>Can constructor be inherited?</td><td>No, constructor cannot be inherited, though a derived class can call the base class constructor.</td></tr>
<tr><td><b>17</b></td><td>What are the differences between Contructors and Methods?</td><td><table>
<tr><th></th><th>Constructors</th><th>Methods</th></tr>
<tr><td>Purpose</td><td>Create an instance of a class</td><td>Group Java statements</td></tr>
<tr><td>Modifiers</td><td>Cannot be abstract, final, native, static, or synchronized</td><td>Can be abstract, final, native, static, or synchronized</td></tr>
<tr><td>Return Type</td><td>No return type, not even void</td><td>void or a valid return type</td></tr>
<tr><td>Name</td><td>Same name as the class (first letter is capitalized by convention) -- usually a noun</td><td>Any name except the class. Method names begin with a lowercase letter by convention -- usually the name of an action</td></tr>
<tr><td>This</td><td>Refers to another constructor in the same class. If used, it must be the first line of the constructor</td><td>Refers to an instance of the owning class. Cannot be used by static methods.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>Super</td><td>Calls the constructor of the parent class. If used, must be the first line of the constructor</td><td>Calls an overridden method in the parent class</td></tr><tr><td>Inheritance</td><td>Constructors are not inherited</td><td>Methods are inherited</td></tr></table></td></tr>
<tr><td><b>18</b></td><td>How are this() and super() used with constructors?</td><td>Constructors use this to refer to another constructor in the same class with a different parameter list.
Constructors use super to invoke the superclass's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.</td></tr>
<tr><td><b>19</b></td><td>What are the differences between Class Methods and Instance Methods?</td><td><table><tr><th>Class Methods</th><th>Instance Methods</th></tr><tr><td>Class methods are methods which are declared as static. The method can be called without creating an instance of the class</td><td>Instance methods on the other hand require an instance of the class to exist before they can be called, so an instance of a class needs to be created by using the new keyword. Instance methods operate on specific instances of classes.</td></tr><tr><td>Class methods can only operate on class members and not on instance members as class methods are unaware of instance members.</td><td>Instance methods of the class can also not be called from within a class method unless they are being called on an instance of that class.</td></tr><tr><td>Class methods are methods which are declared as static. The method can be called without creating an instance of the class.</td><td>Instance methods are not declared as static.</td></tr></table></td></tr>
<tr><td><b>20</b></td><td>How are this() and super() used with constructors?</td><td>Constructors use this to refer to another constructor in the same class with a different parameter list.
Constructors use super to invoke the superclass's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.</td></tr>
<tr><td><b>21</b></td><td>What are Access Specifiers?</td><td>One of the techniques in object-oriented programming is encapsulation. It concerns the hiding of data in a class and making this class available only through methods. Java allows you to control access to classes, methods, and fields via so-called access specifiers..</td></tr>
<tr><td><b>22</b></td><td>What is final modifier?</td><td>The final modifier keyword makes that the programmer cannot change the value anymore. The actual meaning depends on whether it is applied to a class, a variable, or a method.
final Classes- A final class cannot have subclasses.
final Variables- A final variable cannot be changed once it is initialized.
final Methods- A final method cannot be overridden by subclasses.</td></tr>
<tr><td><b>23</b></td><td>What are the uses of final method?</td><td>There are two reasons for marking a method as final:
Disallowing subclasses to change the meaning of the method.
Increasing efficiency by allowing the compiler to turn calls to the method into inline Java code.</td></tr>
<tr><td><b>24</b></td><td>What is method overloading?</td><td>Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type.
Note:
Overloaded methods MUST change the argument list
Overloaded methods CAN change the return type
Overloaded methods CAN change the access modifier
Overloaded methods CAN declare new or broader checked exceptions
A method can be overloaded in the same class or in a subclass</td></tr>
<tr><td><b>25</b></td><td>What is method overriding?</td><td>Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behaviour that’s specific to a particular subclass type.</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Note:
The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can’t override a method marked public and make it protected).
You cannot override a method marked final
You cannot override a method marked static</td></tr>
<tr><td><b>26</b></td><td>What are the differences between method overloading and method overriding?</td><td>
<table>
<tr><th></th><th>Overloaded Method</th><th>Overridden Method</th></tr>
<tr><td>Arguments</td><td>Must change</td><td>Must not change</td></tr>
<tr><td>Return type</td><td>Can change</td><td>Can’t change except for covariant returns</td></tr>
<tr><td>Exceptions</td><td>Can change</td><td>Can reduce or eliminate. Must not throw new or broader checked exceptions</td></tr>
<tr><td>Access</td><td>Can change</td><td>Must not make more restrictive (can be less restrictive)</td></tr>
<tr><td>Invocation</td><td>Reference type determines which overloaded version is selected. Happens at compile time.</td><td>Object type determines which method is selected. Happens at runtime.</td></tr>
</table>
</td></tr>
<tr><td><b>27</b></td><td>Can overloaded methods be override too?</td><td>Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.</td></tr>
<tr><td><b>28</b></td><td>Is it possible to override the main method?</td><td>NO, because main is a static method. A static method can't be overridden in Java.</td></tr>
<tr><td><b>29</b></td><td>How to invoke a superclass version of an Overridden method?</td><td>To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass' implementation of the method.
// From subclass
super.overriddenMethod();</td></tr>
<tr><td><b>30</b></td><td>How do you prevent a method from being overridden?</td><td>To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means "this is the final implementation of this method", the end of its inheritance hierarchy.
public final void exampleMethod() {
// Method statements
}</td></tr>
</table>

<table>
<tr><th>Chapter -4</th><th>Java class Design</th><th></th></tr>
<tr><td><b>1</b></td><td>What are different types of access modifiers?</td><td>Access specifiers are keywords that determine the type of access to the member of a class. These keywords are for allowing
privileges to parts of a program such as functions and variables. These are:
• <i>Public</i> : accessible to all classes
• <i>Protected</i> : accessible to the classes within the same package and any subclasses.
• <i>Private</i> : accessible only to the class to which they belong
• <i>Default :</i> accessible to the class to which they belong and to subclasses within the same package</td></tr>
<tr><td><b>2</b></td><td></td><td>
<table>
<tr><th>Situation</th><th>public</th><th>protected</th><th>default</th><th>private</th></tr>
<tr><td>Accessible to class from same package?</td><td>yes</td><td>yes</td><td>yes</td><td>no</td></tr>
<tr><td>Accessible to class from different</td><td>yes</td><td>no, unless it is a subclass</td><td>no</td><td>no</td></tr>
</table>
</td></tr>
</table>

		package?
3	What is instanceof operator	instanceof keyword to determine an object’s class type at run time.

<table>
<tr><th><strong>Chapter 5</strong></th><th><strong>Advance Class Design</strong></th><th></th></tr>
<tr><td><strong>1</strong></td><td>What is an abstract class?</td><td>Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.
Note:
If even a single method is abstract, the whole class must be declared abstract.
Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.
You can’t mark a class as both abstract and final.</td></tr>
<tr><td><strong>2</strong></td><td>Can we instantiate an abstract class?</td><td>An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).</td></tr>
<tr><td><strong>3</strong></td><td>What are the differences between Interface and Abstract class?</td><td>
<table>
<tr><th>Abstract Class</th><th>Interfaces</th></tr>
<tr><td>An abstract class can provide complete, default code and/or just the details that have to be overridden.</td><td>An interface cannot provide any code at all,just the signature.</td></tr>
<tr><td>In case of abstract class, a class may extend only one abstract class.</td><td>A Class may implement several interfaces.</td></tr>
<tr><td>An abstract class can have non-abstract methods.</td><td>All methods of an Interface are abstract.</td></tr>
<tr><td>An abstract class can have instance variables.</td><td>An Interface cannot have instance variables.</td></tr>
<tr><td>An abstract class can have any visibility: public, private, protected.</td><td>An Interface visibility must be public (or) none.</td></tr>
<tr><td>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.</td><td>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.</td></tr>
<tr><td>An abstract class can contain constructors .</td><td>An Interface cannot contain constructors .</td></tr>
<tr><td>Abstract classes are fast.</td><td>Interfaces are slow as it requires extra indirection to find corresponding method in the actual class.</td></tr>
</table>
</td></tr>
<tr><td><strong>4</strong></td><td>When should I use abstract classes and when should I use interfaces?</td><td>Use Interfaces when…
You see that something in your design will change frequently.
If various implementations only share method signatures then it is better to use Interfaces.
you need some classes to use some methods which you don't want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.
Use Abstract Class when…
If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass.
Abstract classes are an excellent way to create planned inheritance hierarchies. They're also a good choice for nonleaf classes in class hierarchies.</td></tr>
<tr><td><strong>5</strong></td><td>When you declare a</td><td>Yes, other nonabstract methods can access a method that you declare</td></tr>
</table>

	method as abstract, can other nonabstract methods access it?	as abstract.
6	Can there be an abstract class with no abstract methods in it?	Yes, there can be an abstract class without abstract methods.
7	What is static block?	Static block which exactly executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.
8	What are static variables?	Variables that have only one copy per class are known as static variables. They are not attached to a particular instance of a class but rather belong to a class as a whole. They are declared by using the static keyword as a modifier.static type varIdentifier; where, the name of the variable is varIdentifier and its data type is specified by type. Note: Static variables that are not explicitly initialized in the code are automatically initialized with a default value. The default value depends on the data type of the variables.
9	What is the difference between static and non-static variables?	A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.
10	What are static methods?	Methods declared with the keyword static as modifier are called static methods or class methods. They are so called because they affect a class as a whole, not a particular instance of the class. Static methods are always invoked without reference to a particular instance of a class. Note:The use of a static method suffers from the following restrictions: A static method can only call other static methods. A static method must only access static data. A static method cannot reference to the current object using keywords super or this.
11	What are final classes	Classes which caanot be subclasses further are called as final classes.
12	What are Final variables	Final variables are those whose value cannot be modified by any part of the program
13	Can Enum implement interface in Java?	Yes, Enum can implement interface in Java. Since enum is a type, similar to class and interface, it can implement interface. This gives a lot of flexibility to use Enum as specialized implementation in some cases.
14	Can Enum extends class in Java?	No, Enum can not extend class in Java. Surprised, because I just said it's a type like a class or interface in Java. Well, this is why this question is a good follow-up question of previous Enum interview question. Since all Enum by default extend abstract base class java.lang.Enum, obviously they can not extend another class, because Java doesn't support multiple inheritance for classes. Because of extending java.lang.Enum class, all enum gets methods like ordinal(), values() or valueOf().

15	Can we override toString() method for Enum? What happens if we don't?	Ofcourse you can override toString in Enum, as like any other class it also extends java.lang.Object and has toString() method available, but even if you don't override, you will not going to regret much, because abstract base class of enum does that for you and return name, which is name of the enum instance itself. here is the code of toString() method from Enum class :
		
		public String toString() {
		return name;
		}

16	Can we declare Constructor inside Enum in Java?	Yes, you can, but remember you can only declare either private or package-private constructor inside enum. public and protected constructors are not permitted inside enum.

17	Can we use Enum in switch case in Java?	Yes, you can use Enum in Switch case in Java, in fact that's one of the main advantage of using Enum. Since Enum instances are compile time constant, you can safely use them inside switch and case statements. Here is an example of using our DayOfWeek enum in switch case :
		
		public void developerState(DayOfWeek today){
		switch(today){
		case MONDAY:

		System.out.println("Hmmmmmmmm");
		break;
		case TUESDAY:
		System.out.println("Hmmmm");
		break;
		case FRIDAY :
		System.out.println("Yeahhhhhh");
		break;
		}
		}
		Enum and Switch cases goes well with each other, especially if Enum has relatively small number of fixed constants e.g. 7 days in week, 12 months in a year etc, See here for another example of using switch case with Enum in Java

18	How to iterate over all instance of a Enum?	Well, if you have explored java.lang.Enum, you know that there is a values() method which returns an array of all enum constant. Since every enum type implicitly extends java.lang.Enum, they get this values() method. By using, this you can iterate over all enum constants of a certain type. See here for a Enum values Example in Java for iterating over Enum using values() and foreach loop.
19	What is advantage and disadvantage of using Enum as Singleton?	Enum provides you a quick shortcut to implement Singleton design pattern, and ever since it's mentioned in Effective Java, it's been a popular choice as well. On the face, Enum Singleton looks very promising and handles lot of stuff for you e.g. controlled instance creation, Serialization safety and on top of that, it's extremely easy to create thread-safe Singleton using Enum. You don't need to worry about double checked locking and volatile variable anymore. See here to know about pros and cons of using Enum as Singleton in Java.
20	What are inner classes	A inner class is declared to logically group classes and interfaces in one place so that it can be more readable and maintainable. Additionally, it can access all the members of outer class including private data members and methods.

Chapter 6	**Inheritance with Java Interfaces**	
1	Define Interfaces	Java interfaces are used to define abstract types. Interfaces: • Are similar to abstract classes containing only public abstract methods • Outline methods that must be implemented by a class – Methods must not have an implementation {braces}. • Can contain constant fields • Can be used as a reference type • Are an essential component of many design patterns

2	What are Marker Interfaces	Marker interfaces define a type but do not outline any methods that must be implemented by a class.
		public class Person implements java.io.Serializable { }
		• The only reason these type of interfaces exist is type checking.
		Person p = new Person();
		if (p instanceof Serializable) {
		}

3	What is a DAO pattern	The DAO pattern moves the persistence logic out of the domain classes and into separate classes.

4	What is a Factory Pattern	Using a factory prevents your application from being tightly
		coupled to a specific DAO implementation
		EmployeeDAOFactory factory = new EmployeeDAOFactory();
		EmployeeDAO dao = factory.createEmployeeDAO();

Chapter 7	**Generics And Collections**	
1	What are generics?	Provide flexible type safety to your code • Move many common errors from runtime to compile time • Provide cleaner, easier-to-write code • Reduce the need for casting with collections • Are used heavily in the Java Collections API
2	What is Type Inference	With diamond inference there is no need to repeat types on the right side

<table>
<tr><td></td><td>Diamond</td><td>of the statement.
Angle brackets indicate that type parameters are mirrored.It simplifies generic declarations and
Saves typing</td></tr>
<tr><td><b>3</b></td><td>How do you traverse through a collection using its Iterator?</td><td>To use an iterator to traverse through the contents of a collection, follow these steps:
Obtain an iterator to the start of the collection by calling the collectionâ€™s iterator() method.
Set up a loop that makes a call to hasNext(). Have the loop iterate as long as hasNext() returns true.
Within the loop, obtain each element by calling next().</td></tr>
<tr><td><b>4</b></td><td>How do you remove elements during Iteration?</td><td>Iterator also has a method remove() when remove is called, the current element in the iteration is deleted.</td></tr>
<tr><td><b>5</b></td><td>What is the difference between Enumeration and Iterator?</td><td><table>
<tr><th>Enumeration</th><th>Iterator</th></tr>
<tr><td>Enumeration doesn't have a remove() method</td><td>Iterator has a remove() method</td></tr>
<tr><td>Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects</td><td>Can be abstract, final, native, static, or synchronized</td></tr>
</table></td></tr>
<tr><td><b>6</b></td><td>What is the List interface?</td><td>The List interface provides support for ordered collections of objects.Lists may contain duplicate elements</td></tr>
<tr><td><b>7</b></td><td>What are the main implementations of the List interface</td><td>The main implementations of the List interface are as follows :
ArrayList : Resizable-array implementation of the List interface. The best all-around implementation of the List interface.
Vector : Synchronized resizable-array implementation of the List interface with additional "legacy methods."
LinkedList : Doubly-linked list implementation of the List interface. May provide better performance than the ArrayList 8implementation if elements are frequently inserted or deleted within the list. Useful for queues and double-ended queues (deques).</td></tr>
<tr><td><b>8</b></td><td>What are the advantages of ArrayList over arrays ?</td><td>Some of the advantages ArrayList has over arrays are:
It can grow dynamically It provides more powerful insertion and search mechanisms than arrays.</td></tr>
<tr><td><b>9</b></td><td>Difference between ArrayList and Vector ?</td><td><table>
<tr><th>ArrayList</th><th>Vector</th></tr>
<tr><td>ArrayList is NOT synchronized by default.</td><td>Vector List is synchronized by default.</td></tr>
<tr><td>ArrayList can use only Iterator to access the elements.</td><td>Vector list can use Iterator and Enumeration Interface to access the elements.</td></tr>
<tr><td>The ArrayList increases its array size by 50 percent if it runs out of room.</td><td>A Vector defaults to doubling the size of its array if it runs out of room</td></tr>
<tr><td>ArrayList has no default size.</td><td>While vector has a default size of 10.</td></tr>
</table></td></tr>
<tr><td><b>10</b></td><td>How do you decide when to use ArrayList and When to use LinkedList?</td><td>If you need to support random access, without inserting or removing elements from any place other than the end, then ArrayList offers the optimal collection. If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially, then LinkedList offers the better implementation.</td></tr>
<tr><td><b>11</b></td><td>What is the Set interface ?</td><td>The Set interface provides methods for accessing the elements of a finite mathematical set
Sets do not allow duplicate elements
Contains no methods other than those inherited from Collection
It adds the restriction that duplicate elements are prohibited
Two Set objects are equal if they contain the same elements</td></tr>
<tr><td><b>12</b></td><td>What are the main Implementations of the Set interface ?</td><td>The main implementations of the List interface are as follows:
HashSet
TreeSet
LinkedHashSet
EnumSet</td></tr>
</table>

<table>
<tr><th></th><th></th><th></th></tr>
<tr><td><b>13</b></td><td>What is a HashSet ?</td><td>A HashSet is an unsorted, unordered Set.
It uses the hashcode of the object being inserted (so the more efficient your hashcode() implementation the better access performance you’ll get). Use this class when you want a collection with no duplicates and you don’t care about order when you iterate through it.</td></tr>
<tr><td><b>14</b></td><td>What is a TreeSet ?</td><td>TreeSet is a Set implementation that keeps the elements in sorted order. The elements are sorted according to the natural order of elements or by the comparator provided at creation time.</td></tr>
<tr><td><b>15</b></td><td>What is an EnumSet ?</td><td>An EnumSet is a specialized set for use with enum types, all of the elements in the EnumSet type that is specified, explicitly or implicitly, when the set is created.</td></tr>
<tr><td><b>16</b></td><td>Difference between HashSet and TreeSet ?</td><td><table>
<tr><th>HashSet</th><th>TreeSet</th></tr>
<tr><td>HashSet is under set interface i.e. it does not guarantee for either sorted order or sequence order.</td><td>TreeSet is under set i.e. it provides elements in a sorted order (acceding order).</td></tr>
<tr><td>We can add any type of elements to hash set.</td><td>We can add only similar types of elements to tree set.</td></tr>
</table></td></tr>
<tr><td><b>17</b></td><td>What is a Map ?</td><td>A map is an object that stores associations between keys and values (key/value pairs).
Given a key, you can find its value. Both keys and values are objects. The keys must be unique, but the values may be duplicated.
Some maps can accept a null key and null values, others cannot.</td></tr>
<tr><td><b>18</b></td><td>What are the main Implementations of the Map interface ?</td><td>The main implementations of the List interface are as follows:
HashMap
HashTable
TreeMap
EnumMap</td></tr>
<tr><td><b>19</b></td><td>What is a TreeMap</td><td>TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black tree data structure.</td></tr>
<tr><td><b>20</b></td><td>How do you decide when to use HashMap and when to use TreeMap ?</td><td>For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.</td></tr>
<tr><td><b>21</b></td><td>Difference between HashMap and Hashtable ?</td><td><table>
<tr><th>HashMap</th><th>Hashtable</th></tr>
<tr><td>HashMap lets you have null values as well as one null key.</td><td>HashTable does not allows null values as key and value.</td></tr>
<tr><td>The iterator in the HashMap is fail-safe (If you change the map while iterating, you’ll know).</td><td>The enumerator for the Hashtable is not fail-safe.</td></tr>
<tr><td>HashMap is unsynchronized.
Note: Only one NULL is allowed as a key in HashMap. HashMap does not allow multiple keys to be NULL. Nevertheless, it can have multiple NULL values.</td><td>Hashtable is synchronized.</td></tr>
</table></td></tr>
<tr><td><b>21</b></td><td>How does a Hashtable internally maintain the key-value pairs?</td><td>TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black</td></tr>
</table>

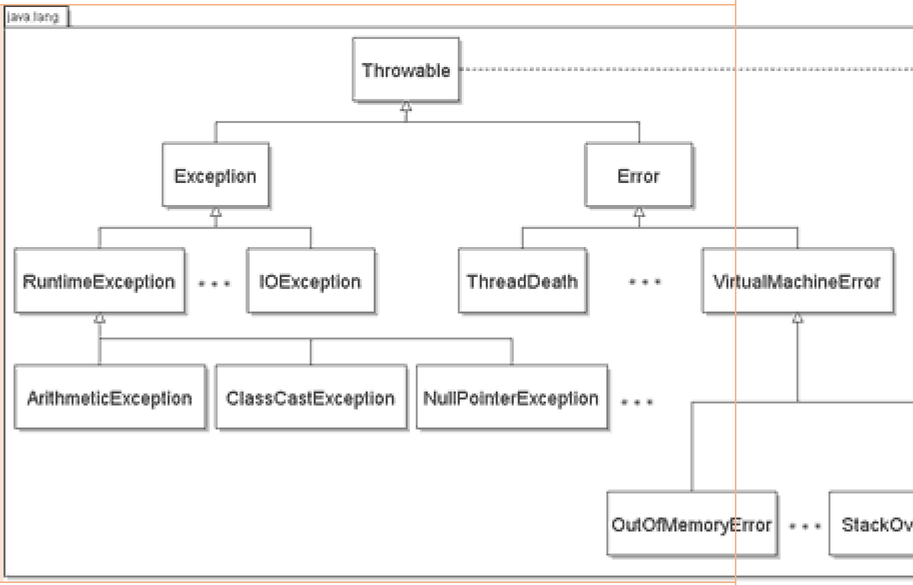
		tree data structure.

Chapter 8	Strings	
1	What is String in Java? String is a data type?	String is a Class in java and defined in java.lang package. It's not a primitive data type like int and long. String class represents character Strings. String is used in almost all the Java applications and there are some interesting facts we should know about String. String in immutable and final in Java and JVM uses String Pool to store all the String objects. Some other interesting things about String is the way we can instantiate a String object using double quotes and overloading of "+" operator for concatenation.
2	How to compare two Strings in java program?	Java String implements Comparable interface and it has two variants of compareTo() methods.compareTo(String anotherString) method compares the String object with the String argument passed lexicographically. If String object precedes the argument passed, it returns negative integer and if String object follows the argument String passed, it returns positive integer. It returns zero when both the String have same value, in this case equals(String str) method will also return true. compareToIgnoreCase(String str): This method is similar to the first one, except that it ignores the case. It uses String CASE_INSENSITIVE_ORDER Comparator for case insensitive comparison. If the value is zero thenequalsIgnoreCase(String str) will also return true.
3	How to convert String to char and vice versa?	This is a tricky question because String is a sequence of characters, so we can't convert it to a single character. We can use use charAt method to get the character at given index or we can use toCharArray()method to convert String to character array.
4	Can we use String in switch case?	This is a tricky question used to check your knowledge of current Java developments. Java 7 extended the capability of switch case to use Strings also, earlier java versions doesn't support this. If you are implementing conditional flow for Strings, you can use if-else conditions and you can use switch case if you are using Java 7 or higher versions.
5	Difference between String, StringBuffer and StringBuilder?	String is immutable and final in java, so whenever we do String manipulation, it creates a new String. String manipulations are resource consuming, so java provides two utility classes for String manipulations – StringBuffer and StringBuilder.StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So when multiple threads are working on same String, we should use StringBuffer but in single threaded environment we should use StringBuilder. StringBuilder performance is fast than StringBuffer because of no overhead of synchronization.
6	Why String is immutable or final in Java	There are several benefits of String because it's immutable and final. String Pool is possible because String is immutable in java. It increases security because any hacker can't change its value and it's used for storing sensitive information such as database username, password etc.Since String is immutable, it's safe to use in multi-threading and we don't need any synchronization. Strings are used in java classloader and immutability provides security that correct class is getting loaded by Classloader
7	Does String is thread-safe in Java?	Strings are immutable, so we can't change it's value in program. Hence it's thread-safe and can be safely used in multi-threaded environment.
8	Why String is popular HashMap key in Java?	Since String is immutable, its hashcode is cached at the time of creation and it doesn't need to be calculated again. This makes it a great candidate for key in a Map and it's processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.
9	Can we compare String using == operator? What is risk?	You can compare String using equality operator but that is not suggested or advised because equality operator is used to compare primitives and equals() method should be used to compare objects. As we have seen in pitfall of autoboxing in Java that how equality operator can cause subtle issue while comparing primitive to Object, any way String is free from that issue because it doesn't have corresponding primitive type and not participate in autoboxing. Almost all the time comparing String means comparing contents of String i.e. characters and equals() method is used to perform character based comparison. equals() return true if two String points to same object or two String has same contents while == operator

		returns true if two String object points to same object but return false if two different String object contains same contents. That explains why sometime it works and sometime it doesn't. In short always use equals method in Java to check equality of two String object.
10	What substring() method does?	In JDK 6, the substring() method gives a window to an array of chars which represents the existing String, but do not create a new one. To create a new array to back string, you can do add an empty string like the following: 1.str.substring(m, n) + ""
11	What is the mean of Regular Expressions ? What the use of Regular Expressions ?	Regular Expressions means we can say a pattern that one is an expression that specifies a set of strings. And there are lot of great use of these expressions like In complex search In validation like email validation or URl validation etc in writing of rewrite rules in apache or .htaccess file in file search in linux in pattern matches
12	What the mean of different symbols like ^, $, * , + , ?, ., {},[] in Regular Expressions	These symbols ^, $, * , + , ?, ., {},[] are very usefull like "^PCDS": Symbols ^ use to matches any string that starts with "PCDS" this can match "PCDS is an Infotech"; "pcds$": Matches a string that ends in the substring "pcds" like "This is pcds"; "^pcds$": Matches a string that starts and ends with "pcds" that could only be "pcds" itself "ab*": matches a string that has an a followed by zero or more b's ("a", "ab", "abbb", etc.); "ab+": same as *, but there's at least one b ("ab", "abbb", etc.); "ab?": there might be a b or not; "ab.": . matches any character like abd, abh, abt etc; "ab{2}": matches a string that has an a followed by exactly two b's ("abb"); "ab{2,}": there are at least two b's ("abb", "abbbb", etc.); "ab{3,5}": from three to five b's ("abbb", "abbbb", or "abbbbb"). "[a-z]": matches any char from a to z.
13	How you can validate email id use REG expression	By using below reg expression we can validate email id ^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$ "^[_a-z0-9-]+": start with any underscore or alphanumeric one or more then one alfanumaric word "(\.[_a-z0-9-]+)*@":any dot(.) or alphanumeric zero times or more the zero and a @ sign "[a-z0-9-]+":one or more alphanumeric "(\.[a-z0-9-]+)*": zero or more dot alphanumeric or -(mins sign) "(\.[a-z]{2,5})$": End with two to 5 character may use dot (.) in that

14	What is the Importance of Pattern.compile()	A regular expression, specified as a string, must first be compiled into an instance of Pattern class. Pattern.*compile()* method is the only way to create a instance of object. A typical invocation sequence is thus
		Pattern p = Pattern.compile("a*b");
		Matcher matcher = p.matcher("aaaaab");
		assert matcher.matches() == true;
		Pattern.*matches()* method is defined as a convenience for when a regular expression is used jus
		instance of a Pattern implicitly, and matches a string. Therefore,
		boolean b = Pattern.matches("a*b", "aaaaab");

Chapter 9	Exceptions And Assertions	
1	What is an Exception.	The exception is said to be thrown whenever an exceptional event occurs in java which signals that something is not correct with the code written and may give unexpected result. An exceptional event is a occurrence of condition which alters the normal program flow. Exceptional handler is the code that does something about the exception.
2	Exceptions are defined in which java package?	All the exceptions are subclasses of java.lang.Exception
3	Difference between throw and throws	The throw key word is used to explicitly throw an exception, while throws is utilized to handle checked exceptions for re-intimating the compiler that exceptions are being handled. It is kind of communicating to the compiler that an exception is expected to throw one or more checked exceptions. The throws need to be used in the method's definition and also while invoking the method that raises checked exceptions.

4	**What is Runtime Exception or unchecked exception?**	Runtime exceptions represent problems that are the result of a programming problem. Such problems include arithmetic exceptions, such as dividing by zero; pointer exceptions: such as trying to access an object through a null reference; and indexing exceptions: such as attempting to access an array element through an index that is too large or too small. Runtime exceptions need not be explicitly caught in try catch block as it can occur anywhere in a program, and in a typical one they can be very numerous. Having to add runtime exceptions in every method declaration would reduce a program's clarity. Thus, the compiler does not require that you catch or specify runtime exceptions (although you can). The solution to rectify is to correct the programming logic where the exception has occurred or provide a check
5	What is checked exception?	Checked exception are the exceptions which forces the programmer to catch them explicitly in try-catch block. It is a subClass of Exception. Example: IOException.
6	How are the exceptions handled in java?	When an exception occurs the execution of the program is transferred to an appropriate exception handler. The try-catch-finally block is used to handle the exception. The code in which the exception may occur is enclosed in a try block, also called as a guarded region. The catch clause matches a specific exception to a block of code which handles that exception. And the clean up code which needs to be executed no matter the exception occurs or not is put inside the finally block
7	Explain the exception hierarchy in java.	 java.lang Throwable Exception Error RuntimeException IOException ThreadDeath VirtualMachineError ArithmeticException ClassCastException NullPointerException OutOfMemoryError StackOv

8	What is Exception chaining in Java?	One of the important exception handling concept in Java when a different exception is thrown in response to an actual exception. This in turn will create a chain of exceptions. This is commonly used to wrap a checked exception to unchecked exception under various scenarios and such coding prevails when actually writing a framework in which case a known checked exception is thrown as an unchecked framework exception. Even though new exception is thrown, as a best practise always include the root cause (exception) in the newly created exception class.
		With regards to JDK7 there is a new feature with regards to Exception handling. Can you detail this?
		There are numerous small and medium new features on JDK7 which are very useful. Some of them with regards to error and exception handling is as detailed below:-
		Multiple exception handling in one catch block
		`public static void main(String args[]) {`
		`    //...Code`
		`    try {`
		`        //..Code`
		`    } catch (NumberFormatException | IllegalArgumentException e) {`
		`            e.printStackTrace();`

		} }

9	ARM (Automatic Resource Management) blocks/explain try with resource	As the name suggests it is capable of automatically handling resources on behalf of you, also known popularly as try with resource.
		static String readFirstLineFromFile(String path) throws IOException {
		try (BufferedReader br =
		new BufferedReader(new FileReader(path))) {
		return br.readLine();
		}
		}

Chapter 10	IO	
1	What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?	The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented.
2	What is synchronization and why is it important?	Serialization is a mechanism by which you can save or transfer the state of an object by converting it to a byte stream. This can be done in java by implementing Serialiazable interface. Serializable is defined as a marker interface which needs to be implemented for transferring an object over a network or persistence of its state to a file. Since its a marker interface, it does not contain any methods. Implementation of this interface enables the conversion of object into byte stream and thus can be transferred. The object conversion is done by the JVM using its default serialization mechanism.
3	What class allows you to read objects directly from a stream?	The ObjectInputStream class supports the reading of objects from input streams
4	7. What is the difference between the File and RandomAccessFile classes?	The File class encapsulates the files and directories of the local file system. The RandomAccessFile class provides the methods needed to directly access data contained in any part of a file.
5	What is Serialization and deserialization?	Serialization is the process of writing the state of an object to a byte stream. Deserialization is the process of restoring these objects.
6	What is the purpose of the File class?	The File class is used to create objects that provide access to the files and directories of a local file system.

7	What are transient variables? What role do they play in Serialization process?	The transient keyword in Java is used to indicate that a field should not be serialized. Once the process of de-serialization is carried out, the transient variables do not undergo a change and retain their default value. Marking unwanted fields as transient can help you boost the serialization performance. Below is a simple example where you can see the use of transient keyword.
		class MyVideo implements Serializable
		{
		private Video video;
		private transient Image thumbnailVideo;
		private void generateThumbnail()
		{
		// Generate thumbnail.
		}
		private void readObject(ObjectInputStream inputStream)
		throws IOException, ClassNotFoundException
		{
		inputStream.defaultReadObject();
		generateThumbnail();
		}
		}

8	In System.out.printLn(), What is System, out and printLn?	System is a predefined class, out is a PrintStream that represents standard out and printLn is a built-in method for printing output one line at a time.
9	What is an IOException? Can you give some examples of a few sub-classes of this Exception?	IOException signals that an I/O operation of some type has failed. Examples of IOException sub-classes are, FileNotFoundException, EOFException and SocketException.

10	Java Scanner class	There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.The Java Scanner **class** breaks the input into tokens using a delimiter that is whitespace bydefault. It provides many methods to read and parse various primitive values.Java Scanner class is widely used to parse text for string and primitive types using regular expression. Java Scanner class extends Object class and implements Iterator and Closeable interfaces.
11	What is Console class in Java	The **Java.io.Console** class provides methods to access the character-based console device, if any, associated with the current Java virtual machin

Chapter 11	NIO	
1	Describe the usage of java.nio.channels in java.	java.nio.channels defines the channels and selectors. Channels are used for entities that can perform I/O operations. Selectors are used for non blocking I/O operations.
2	What are differences between SocketChannel and ServerSocketChannel in java?	**a.** SocketChannel is selectable channel for stream-oriented connecting sockets whereas ServerSocketChannel is selectable channel for stream-oriented listening socket. **b.** SocketChannel and ServerSocketChannel both are created by invoking its open method. **c.** SocketChannel and ServerSocketChannel both are thread safe.

Chapter 12		
1	What is a thread? What are the advantages we derived by programming with thread?	Threads allow programs to execute simultaneously. A thread is an independent path of execution in a program. These threads can be executed synchronously or asynchronously. All threads have a priority attached. Thread with a higher priority is executed first. Advantages: Allows multiple programs to be executed concurrently Cost of thread is low compared to processes in terms of space and communication. Threads are lightweight.
2	What is difference between a thread and a process?	Threads share the address space of the process that created it; processes have their own address. 2. Threads have direct access to the data segment of its process; processes have their own copy of the data segment of the parent process. 3. Threads can directly communicate with other threads of its process; processes must use interprocess communication to communicate with sibling processes. 4. Threads have almost no overhead; processes have considerable overhead. 5. New threads are easily created; new processes require duplication of the parent process. 6. Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes. 7. Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process do not affect child processes

3	What are the two ways of creating thread?	There are two ways to create a new thread.
		Extend the Thread class and override the run() method in your class. Create an instance of the subclass and invoke the start() method on it, which will create a new thread of execution.
		public class NewThread extends Thread{
		  public void run(){
		    // the code that has to be executed in a separate new thread goes here
		    }
		    public static void main(String [] args){
		    NewThread c = new NewThread();
		    c.start();
		  }
		}
		Implements the Runnable interface.The class will have to

		implement the run() method in the Runnable interface. Create an instance of this class. Pass the reference of this instance to the Thread constructor a new thread of execution will be created.
		public class NewThread implements Runnable{
		    public void run(){
		      // the code that has to be executed in a separate new thread goes here
		      }
		    public static void main(String [] args){
		      NewThread c = new NewThread();
		      Thread t = new Thread(c);
		      t.start();
		     }
		    }

4	What states can a thread have and what is the meaning of each state?	NEW: A thread that has not yet started is in this state. RUNNABLE: A thread executing in the Java virtual machine is in this state. BLOCKED: A thread that is blocked waiting for a monitor lock is in this state. WAITING: A thread that is waiting indefinitely for another thread to perform a particular action is in this state. TIMED_WAITING: A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state. TERMINATED: A thread that has exited is in this state.
5	How do we set the priority of a thread?	The priority of a thread is set by using the method setPriority(int). To set the priority to the maximum value, we use the constant Thread.MAX_PRIORITY and to set it to the minimum value we use the constant Thread.MIN_PRIORITY because these values can differ between different JVM implementations.

6	For what purposes is the keyword synchronized used?	When you have to implement exclusive access to a resource, like some static value or some file reference, the code that works with the exclusive resource can be embraced with a synchronized block:
		synchronized (SynchronizedCounter.class) {
		    counter++;
		}

Chapter 13	**Concurrency**	
1	What do you mean by concurrency	Concurrency is the ability of a program to execute several computations simultaneously. This can be achieved by distributing the computations over the available CPU cores of a machine or even over different machines within the same network.
2	What is a scheduler?	A scheduler is the implementation of a scheduling algorithm that manages access of processes and threads to some limited resource like the processor or some I/O channel. The goal of most scheduling algorithms is to provide some kind of load balancing for the available processes/threads that guarantees that each process/thread gets an appropriate time frame to access the requested resource exclusively.
3	What operations are atomic in Java?	The Java language provides some basic operations that are atomic and that therefore can be used to make sure that concurrent threads always see the same value: Read and write operations to reference variables and primitive variables (except long and double) Read and write operations for all variables declared as volatile
4	What do we understand by a deadlock?	A deadlock is a situation in which two (or more) threads are each waiting on the other thread to free a resource that it has locked, while the thread itself has locked a resource the other

		thread is waiting on: Thread 1: locks resource A, waits for resource B Thread 2: locks resource B, waits for resource A
5	What is Thread Pools ?	Most of the executor implementations in java.util.concurrent use thread pools, which consist of worker threads. This kind of thread exists separately from the Runnable and Callable tasks it executes and is often used to execute multiple tasks.Using worker threads minimizes the overhead due to thread creation. Thread objects use a significant amount of memory, and in a large-scale application, allocating and deallocating many thread objects creates a significant memory management overhead.One common type of thread pool is the fixed thread pool. This type of pool always has a specified number of threads running; if a thread is somehow terminated while it is still in use, it is automatically replaced with a new thread. Tasks are submitted to the pool via an internal queue, which holds extra tasks whenever there are more active tasks than threads. An important advantage of the fixed thread pool is that applications using it degrade gracefully. To understand this, consider a web server application where each HTTP request is handled by a separate thread. If the application simply creates a new thread for every new HTTP request, and the system receives more requests than it can handle immediately, the application will suddenly stop responding to all requests when the overhead of all those threads exceed the capacity of the system. With a limit on the number of the threads that can be created, the application will not be servicing HTTP requests as quickly as they come in, but it will be servicing them as quickly as the system can sustain. A simple way to create an executor that uses a fixed thread pool is to invoke the newFixedThreadPool factory method in java.util.concurrent.ExecutorsThis class also provides the following factory methods: • The ***newCachedThreadPool*** method creates an executor with an expandable thread pool. This executor is suitable for applications that launch many short-lived tasks. • The ***newSingleThreadExecutor*** method creates an executor that executes a single task at a time. • Several factory methods are ***ScheduledExecutorService*** versions of the above executors. If none of the executors provided by the above factory methods meet your needs, constructing instances of java.util.concurrent.ThreadPoolExecutor or java.util.concurrent.ScheduledThreadPoolExecutor will give you additional options.
6	What is Fork/Join ?	New in the Java SE 7 release, the fork/join framework is an implementation of the ExecutorService interface that helps you take advantage of multiple processors. It is designed for work that can be broken into smaller pieces recursively. The goal is to use all the available processing power to make your application wicked fast. As with any ExecutorService, the fork/join framework distributes tasks to worker threads in a thread pool. The fork/join framework is distinct because it uses a work-stealing algorithm. Worker threads that run out of things to do can steal tasks from other threads that are still busy. The center of the fork/join framework is the ForkJoinPool class, an extension of AbstractExecutorService. ForkJoinPool implements the core work-stealing algorithm and can execute ForkJoinTasks.
7	What is Executors ?	In large-scale applications, it makes sense to separate thread management and creation from the rest of the application. Objects that encapsulate these functions are known as executors. The following subsections describe executors in detail. • Executor Interfaces define the three executor object types.

		• Thread Pools are the most common kind of executor implementation. • Fork/Join is a framework (new in JDK 7) for taking advantage of multiple processors. Executor Interfaces The java.util.concurrent package defines three executor interfaces: • Executor, a simple interface that supports launching new tasks. • ExecutorService, a subinterface of Executor, which adds features that help manage the lifecycle, both of the individual tasks and of the executor itself. • ScheduledExecutorService, a subinterface of ExecutorService, supports future and/or periodic execution of tasks. Typically, variables that refer to executor objects are declared as one of these three interface types, not with an executor class type.
8	What is Reentrant Synchronization ?	Recall that a thread cannot acquire a lock owned by another thread. But a thread can acquire a lock that it already owns. Allowing a thread to acquire the same lock more than once enables reentrant synchronization. This describes a situation where synchronized code, directly or indirectly, invokes a method that also contains synchronized code, and both sets of code use the same lock. Without reentrant synchronization, synchronized code would have to take many additional precautions to avoid having a thread cause itself to block.

Chapter 14	**JDBC**	
1	What are the main components of JDBC ?	DriverManager: Manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication subprotocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection. Driver: The database communications link, handling all communication with the database. Normally, once the driver is loaded, the developer need not call it explicitly. Connection : Interface with all methods for contacting a database.The connection object represents communication context, i.e., all communication with database is through connection object only. Statement : Encapsulates an SQL statement which is passed to the database to be parsed, compiled, planned and executed. ResultSet: The ResultSet represents set of rows retrieved due to query execution.
1	What is the functions of Connection,Statement,Resultset?	1.Connection is an interface that provide a session with database. 2.Statement is an interface that provide a class for executing SQL statement and returning a resultset. 3.Resultset ia an interface that manage a resulting data returned from a statement.
2	What are the three methods for writing a queries?	1.executeQuery(sqlString):- For a SELECT statement returns a resultset object. 2.executeUpdate(sqlString):-For INSERT,UPDATE and DELETE statement returns an int. 3.execute(sqlString):-For any Sql Statement returns boolean indicating if a resultset was returned.
3	Which type of JDBC driver is the fastest one?	JDBC Net pure Java driver(Type 4) is the fastest driver because it converts the JDBC calls into vendor specific protocol calls and it directly interacts with the database.
4	Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?	No. You can open only one Statement object per connection when you are using the JDBC-ODBC Bridge.
5	How cursor works in scrollable result set?	There are several methods in the ResultSet interface that involve moving the cursor, like: beforeFirst(), afterLast(), first(), last(), absolute(int row), relative(int row), previous(), next(), getRow(), moveToInsertRow(), moveToCurrentRow().

6	What is ResultsetMetaData	
7	What is a transaction?	A tansaction is a mechanism which handles group of operation as through they were one.
8	What is ACID property of transaction?	1.Atomicity:- All or nothing.All operation invovlved in the transaction are implemented or none one. 2.Consistancy:-The database must be modified from one consistant state from another. 3.Isolation:- An execute transaction is isolated from another executing transaction in terms of database record it is accessing. 4.Durability:-After a transaction is commited, it can be restored to this state in the event os system or database failure.
9	When a connection is created in JDBC Transaction what will be the default mode?	Auto-Commit.
10	How to disable auto-commit mode?	con.setAutoCommit(false);
11	Which statement method execute a SQL statement and returns the no of rows affected?	stmt.executeUpdate(query);
12	which rowset type provides a container for rows of data that caches its rows in memory?	CachedRowSet

	How will you insert multiple rows into a database in a single transaction?	Follow steps as below:
		//turn off the implicit commit
		Connection.setAutoCommit(false);
		//..your insert/update/delete goes here
		Connection.Commit();
		a new transaction is implicitly started.

13	Why do you have to close database connections in Java?	You need to close the resultset, the statement and the connection. If the connection has come from a pool, closing it actually sends it back to the pool for reuse. We can do this in the finally{} block, such that if an exception is thrown, you still get the chance to close this.
14	What we set the attribute Concurrency in ResultSet?	The ResultSet concurrency determines whether the ResultSet can be updated, or only read. A ResultSet can have one of two concurrency levels: ResultSet.CONCUR_READ_ONLY :means that the ResultSet can only be read. ResultSet.CONCUR_UPDATABLE : means that the ResultSet can be both read and updated.
15	What do you mean by cold backup, hot backup?	Cold back is the backup techniques in which backup of files are taken before the database restarted. In hot backup backup of files and table is taken at the same time when database is running. A warm is a recovery technique where all the tables are locked and users cannot access at the time of backing up data.
16	What are the types of JDBC Driver Models and explain them?	There are two types of JDBC Driver Models and they are: a) Two tier model b) Three tier model Two tier model: In this model, Java applications interact directly with the database. A JDBC driver is required to communicate with the particular database management system that is being accessed. SQL statements are sent to the database and the results are given to user. This model is referred to as client/server configuration where user is the client and the machine that has the database is called as the server. Three tier model: A middle tier is introduced in this model. The functions of this model are: a) Collection of SQL statements from the client and handing it over to the database, b) Receiving results from database to the client and c) Maintaining control over accessing and updating of the above.
17	What are the types of statements in JDBC?	Statement -- To be used createStatement() method for executing single SQL statement PreparedStatement -- To be used preparedStatement() method for executing same SQL statement over and over . CallableStatement -- To be used prepareCall( ) method for

		multiple SQL statements over and over.
18	How to retrieve the information about the database ?	We can retrieve the info about the database by using inerface java.sql.DatabaseMetaData We can get this object by using getMetaData() method in Connection interface.
19	what are the Different types of exceptions in jdbc?	BatchUpdateException DataTruncation SQLException SQLWarning
20	What is difference between Statement, PreparedStatement and CallableStatement in Java?	One of the classical JDBC interview question. Main difference between Statement and PreparedSatement is performance and avoiding SQL Injection as we have seen in Benefits of using PreparedStatement in Java. While CallableStatement has very specific use in JDBC and used to call stored procedure from Java program
21	What is the difference between execute, executeQuery, executeUpdate?	boolean execute(): Executes the any kind of SQL statement ResultSet executeQuery(): This is used generally for reading the content of the database. The output will be in the form of ResultSet. Generally SELECT statement is used. int executeUpdate(): This is generally used for altering the databases. Generally DROP TABLE or DATABASE, INSERT into TABLE,UPDATE TABLE, DELETE from TABLE statements will be used in this. The output will be in the form of int which denotes the number of rows affected by the query.
22	What is 2 phase commit?	This is one of the most popular JDBC Interview question and asked at advanced level, mostly to senior Java developers on J2EE interviews. Two phase commit is used in distributed environment where multiple process take part in distributed transaction process.In simple word we can understand like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other. In two phase commit, commit or rollback is done by two phases: 1.Commit request phase: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are "yes" then they go ahead for next phase. And if "No "then rollback is performed. 2.Commit phase: according to vote if all the votes are yes then commit is done.
23	What is JDBC Driver interface?	The JDBC Driver interface provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each vendor driver must provide implementations of the java.sql.Connection,Statement,PreparedStatement, CallableStatement, ResultSet and Driver.
24	What is a DataSource?	DataSource object is the representation of a data source in the Java programming language. In basic terms, A DataSource is a facility for storing data. DataSource can be referenced by JNDI. Data Source may point to RDBMS, file System , any DBMS etc..

Chapter -15	Internationalizationalization and Localization	
1	What is Internationalization	Internationalization is also abbreviated as I18N because there are total 18 characters between the first letter 'I' and the last letter 'N'.Internationalization is a mechanism to create such an application that can be adapted to different languages and regions.Internationalization is one of the powerful concept of java if you are developing an application and want to display messages, currencies, date, time etc. according to the specific region or language.
2	What is localization	Localization is also abbreviated as I10N because there are total 10 characters between the first letter 'L' and last letter 'N'. Localization is the mechanism to create such an application that can be adapted to a specific language and region by adding locale-specific text and component.
3	What are resource bundles	Java provides internationalization (i18n) support through resource bundles. For making your application support

		internationalization, you need to create locale specific properties file. The file names follow the pattern of bundle name with language code and country code, for exampleApplicationMessages_en_US.properties. Once the property files for specific locales are ready, all you need to do it initialize the resource bundle with correct Locale. Java provides two classes java.util.ResourceBundle and java.util.Locale that are used for this purpose. ResourceBundle reads the locale specific property file and you can get the locale specific value for any key. This is very helpful in making your web application texts locale specific, you can get the locale information from the HTTP request and generate the dynamic page with that locale resource bundle files. You can also provide option to user to chose the locale and update the labels dynamically.

4	Formatting Date and Time.	Formatting dates with the DateFormat class is a two-step process.
		1. First, you create a formatter with the getDateInstance method.
		2. Second, you invoke the format method, which returns a String containing the formatted date.
		The following example formats today's date by calling these two methods:
		Date today;
		String dateOut;
		DateFormat dateFormatter;
		dateFormatter = DateFormat.getDateInstance(DateFormat.DEFAULT, currentLocale);
		today = new Date();
		dateOut = dateFormatter.format(today);
		System.out.println(dateOut + " " + currentLocale.toString());

5	How to format Time	Date objects represent both dates and times. Formatting times with the DateFormat class is similar to formatting dates, except that you create the formatter with the getTimeInstance method, as follows: DateFormat timeFormatter = DateFormat.getTimeInstance(DateFormat.DEFAULT, currentLocale);
6	Briefly discuss the steps for localization	Step 1: create the properties file. step 2. Define the Locale. Step 3: step 3. Create a ResourceBundle