1.	**What is Annotation?**
	Annotations are used for providing description to package,class ,interface,enum, constructor, attribute, method, local variable.
	Inside annotation all methods can be declared methods, no method can be implemented one.
	Syntax :
	@interface interfaceName{
	method1-declaration
	method2-declaration
	…
	methodn-declaration
	}
	Each method declaration defines an element of the annotation type.
	Method declarations must not have any parameters or a throws clause.
	Return types are restricted to primitives, String, Class, enums, annotations, and arrays of the preceding types.
	@interfaceName is the name of annotation using which we can provide information.
	E.g. : @interface Test
	{
	String message();
	boolean flag();
	}
	To use the above defined annotation : @Test(message = " _____",flag = true/false)
	You can encapsulate any no. of method declaration in same annotation. Wherever that annotation is used there an all you have to provide value to all the methods by using comma as a delimiter.

2.	**How can I provide a default value for any method in the annotation?** To provide the default value for method of an annotation while declaring it in the annotation design provide default value using default keyword with value.
3.	**Is it compulsory to provide default value to all the methods in an annotation?** default is optional its not at all mandatory in an annotation.
4.	**Why it is advisable to provide default in an annotation?** It is advisable to provide a default so while using an annotation if you are not providing value then it will be using default value.
5.	**If an annotation named Test is defined with 2 methods : String meth1() , String meth2() default "abc". @Test(meth2 = "xyz"). Will it work?** No. It will result in compile time error. As meth1 doesn't have any default value so need to be specified while using it.
6.	**If an annotation named Test is defined with 2 methods : String meth1() , String meth2() default "abc". @Test(meth1 = "xyz"). Will it work?** Yes. It will compile successfully. meth1 → "xyz" , meth2 →"abc"
7.	**If an annotation named Test is defined with 2 methods : String meth1() , String meth2() default "abc".** **@Test(meth1 = "abc",meth2 = "pqr"). Will it work?** Yes. It will compile successfully , meth1→"abc", meth2 →"pqr" (as it is specified explicitly so default will not be taken)
8.	**What is meta annotation? Name any 3 meta annotations.** An annotation used to describe an annotation is called as a meta annotation. @Target, @Retention, @Inherited
9.	**Explain about @Target meta annotation.** One of the built-in annotation available in JDK. Use this annotation to restrict the usage of an annotation on certain java elements only. After

	specifying the targets, you will be able to use the new annotation on given elements only Syntax : @Target(ElementType.element) **element:** CONSTRUCTOR(Constructor declaration) FIELD(Field declaration(including enum constant)) LOCAL_VARIABLE(Local variable declaration) METHOD(Method declaration) PACKAGE(Package declaration) PARAMETER(Parameter declaration) TYPE(Class, Interface(including annotation type), or enum declaration)
10.	**Explain @Inherited meta annotation.** One of the built-in annotation available in JDK. Annotations of super class will be inherited to subclass.
11.	**Explain @Retention meta annotation.** one of the built-in meta-annotation. @Retention annotation defines how the annotation marked with it will be stored using RetentionPolicy. RetentionPolicy.SOURCE →Annotation description will be available in source it won't be in .class file. It is only for the developers. It is not for the compiler or runtime. It will not be considered by the compiler. RetentionPolicy.CLASS → Annotation description will also go in .class file. RetentionPolicy.RUNTIME → Annotation description will go in .class file, while loading class in memory annotation information will also be loading to memory while running.
12.	**How to read available annotations for particular class?** Annotation[] ann = ClassName.class.getAnnotations()
13.	**Discuss about @Override annotation.** One of the built-in annotation. It can be used for overridden method or super class method is being implemented in the subclass. It can't be used for newly incorporated methods.
14.	**Discuss about @Deprecated annotation.** It is to mark an element as deprecated and no longer be used. Deprecated means not advisable but still you can use. While using deprecated methods compiler gives a warning. But still you can run the program successfully.
15.	**Tell something about @SuppressWarnings annotation.** By using this annotation we are telling the compiler don't provide warning related to specify type of warning. @SuppressWarning(value = "type") It can be used on top of class/ method. If it is defined on top of class then will be applicable to all the methods of a class.
16.	**How to know details about the compile time warning?** Using –Xlint:warningType .
17.	**How to suppress multiple types of warning?** @SuppressWarnings({"warningType1","warningType2",…"warningTypen"})
18.	**In which version of JDK Annotation is introduced?** JDK1.5