**Process of redefining method of super class inside sub class is called as  
method overriding.**

When such a super class ref is used to invoke the overriding method then the method to send for execution  
that decision is taken by JRE & not by compiler.

Interface  
• In Java, an interface is a blueprint or template of a class. It is much similar to the Java class but the only  
difference is that it has abstract methods and static constants.

An interface in Java is a mechanism which we mainly use to achieve abstraction and multiple inheritances in Java

We can implement multiple Java Interfaces by a Java class. All methods of an interface are implicitly public and  
abstract. The word abstract means these methods have no method body, only method signature

An interface can inherit or extend multiple interfaces.  
• We can implement more than one interface in our class

Intterface vs class

Unlike a class, you cannot instantiate or create an object of an interface.  
• All the methods in an interface should be declared as abstract.

**Innterface**

* Set of rules are called specification/standard.  
  • It is a contract between service consumer and service provider.
* Interface is non primitive type which helps developer:  
  1. To build/develop trust between service provider and service consumer.  
  2. To minimize vendor dependency.  
  • interface is a keyword in Java.
* Interface  
  During inheritance if supertype and subtype in class then it is called implementation interface  
  1. Single Inheritance( Valid in Java)  
  2. Multiple Inheritance( Valid in Java)  
  3. Hierarchical Inheritance( Valid in Java)  
  4. Multilevel Inheritance( Valid in Java)

Implementation Inheritance  
o During inheritance if super type and sub type  
implementation inheritance.  
1. Single Inheritance( Valid in Java)  
2. Multiple Inheritance( Invalid in Java)  
3. Hierarchical Inheritance( Valid in Java)  
4. Multilevel Inheritance( Valid in Java)

Abstract Class  
  
1. If "is-a" relationship is exist between super type and sub type and if we  
sub types then super type must bewant same method design in all the  
abstract.  
2. Using abstract class, we can group instances of related type together  
only one abstract/concrete class.3. Abstract class can extend  
4. We can define constructor  
5. Abstract class may or may  
inside abstract class.  
not contain abstract method.  
• Hint : In case of inheritance if state is involved in super type then it  
should be abstract.

Interface  
Sunbeam www.sunbeaminfo.com  
1. If "is-a" relationship is not exist between super type and sub type and  
if  
be  
we want same method design  
interface.  
in all the sub types then super type must  
2. Using interface, we can group instances of unrelated type together.  
3. Interface can extend more than one interfaces.  
4. We can not define constructor inside interface.  
5. By default methods of interface are abstract.  
• Hint : In case of inheritance if state is not involved in super type then  
it should be interface.

In the context of exception handling, any class which implements  
java.lang.AutoCloseable or its sub interface( e.g. java.io.Closeable ) is  
called resource type and its instance is called as resource.

Types Of Exception  
Sunbeam www.sunbeaminfo.com  
**• Unchecked Exception**  
o java.lang.RuntimeException and all its sub classes are considered as unchecked  
exception.  
o It is not mandatory to handle unchecked exception.  
o Example:  
1. NullPointerException  
2. ClassCastException  
3. ArrayIndexOutOfBoundsException  
o During the execution of arithmetic operation, if any  
then JVM throws ArithmeticException.  
**• Checked Exception**  
exceptional situation occurs  
o java.lang.Exception and all its sub classes except java.lang.RuntimeException are  
considered as checked exception.  
o It is mandatory to handle checked exception.  
o Example:  
1. java.lang.CloneNotSupportedException  
2. java.lang.InterruptedException

Exception Handling  
  
**• try**  
o It is a keyword in Java.  
o If we want to keep watch on statements for the exception then  
put all such statements inside try block/handler.  
o try block must have at least one:  
1. catch block or  
2. finally block or  
3. Resource  
o We can not define try block after catch or finally block.

**Catch**  
o It is a keyword in Java.  
o If we want to handle exception then we should use catch block/handler  
o Only Throwable class or one of its subclasses can be the argument type in  
a catch clause.  
o Catch block can handle exception thrown from try block only.  
o For single try block we can define multiple catch block.  
o Multi-catch block allows us to handle multiple specific exception inside  
single catch block.

**throw**  
o It is a keyword in Java.  
o If we want to generate new exception then we should use throw keyword.  
o Only objects that are instances of Throwable class (or one of its  
subclasses) are thrown by the Java Virtual Machine or can be thrown by  
the Java throw statement.  
o throw statement is a jump statement.

Exception Handling  
Sunbeam www.sunbeaminfo.com  
**• finally**  
o It is a keyword in Java.  
o If we want to release local resources then we should use finally block.  
o We can not define finally block before try and catch block.

**throws**  
o It is a keyword in Java.  
o If we want to redirect/delegate exception from one method to another then  
we should use throws clause.  
o Consider declaration of following methods:  
1. public static int parseInt(String s) throws NumberFormatException  
2. public static void sleep(long millis) throws InterruptedException

**try-with-resources**  
o The try-with-resources statement is a try statement that declares one or more  
resources.  
o A is an object that must be closed after the program is finished with  
it.  
o The try-with-resources statement ensures that each  
of the statement.  
o Any object that implements java.lang.AutoCloseable,  
which implement java.io.Closeable, can be used as a  
resource is closed at the end  
which includes all objects  
resource

**Marker Interface**  
Sunbeam www.sunbeaminfo.com  
• An interface which do not contain any member is called marker  
interface. In other words, empty interface is called as marker  
interface.  
• Marker interface is also called as tagging interface.  
• If we implement marker interface then Java compiler generates  
metadata for the JVM, which help JVM to clone/serialize or  
marshal state of object.  
• Example:  
1. java.lang.Cloneable  
2. java.util.EventListener  
3. java.util.RandomAccess  
4. java.io.Serializable  
5. java.rmi.Remote

**Generics**

Code is said to be generic if same code can be used for various (practically all) types.

Box<String> b1 = new Box<String>();  
b1.set("Nilesh");  
String obj1 = b1.get();  
System.out.println("obj1 : " + obj1);  
Box<Date> b2 = new Box<Date>();  
b2.set(new Date());  
Date obj2 = b2.get();  
System.out.println("obj2 : " + obj2);  
Box<Integer> b3 = new Box<Integer>();  
b3.set(new Integer(11));  
String obj3 = b3.get(); // Compiler Error  
System.out.println("obj3 : " + obj3);

Instantiating generic class  
Box<String> b1 = new Box<String>(); // okay  
Box<String> b2 = new Box<>(); // okay -- type inference  
Box<> b3 = new Box<>(); // compiler error  
Box<Object> b4 = new Box<String>(); // compiler error  
Box b5 = new Box(); // okay -- if generic type not mentioned, raw type (Object) is considered -- but raise a warning  
Box<Object> b6 = new Box<Object>(); // okay -- can store object of any type -- not usually needed

Advantages of Generics  
Stronger type checking at compile time i.e. type-safe coding.  
Explicit type casting is not required.  
Generic data structure and algorithm implementation.