Java Bean

A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications.

Following are the unique characteristics that distinguish a JavaBean from other Java classes −

- It provides a default, no-argument constructor.
- It should be serializable and that which can implement the Serializable interface.
- It may have a number of properties which can be read or written.
- It may have a number of "getter" and "setter" methods for the properties.

Why use Java Bean?

Java Bean is a reusable software component. A bean encapsulates many objects into one object, so we can access this object from multiple places. Moreover, it provides the easy maintenance.

Simple example of java bean class

```java
//Employee.java
package mypack;  //Used to create a package and is optional
public class Employee implements java.io.Serializable{
private int id;
private String name;
public Employee()
{
}
public void setId(int id){
this.id=id;
}
public int getId(){
return id;
}
public void setName(String name){
this.name=name;
}
public String getName(){
return name;
}
}
```

How to access the java bean class?

To access the java bean class, we should use getter and setter methods.

```java
package mypack;
public class Test{
public static void main(String args[]){
Employee e=new Employee();//object is created
e.setName("Arjun");//setting value to the object
System.out.println(e.getName());
}

}
```

Q. What is Serialization in Java? When is it necessary to implement a class by Serializable interface?
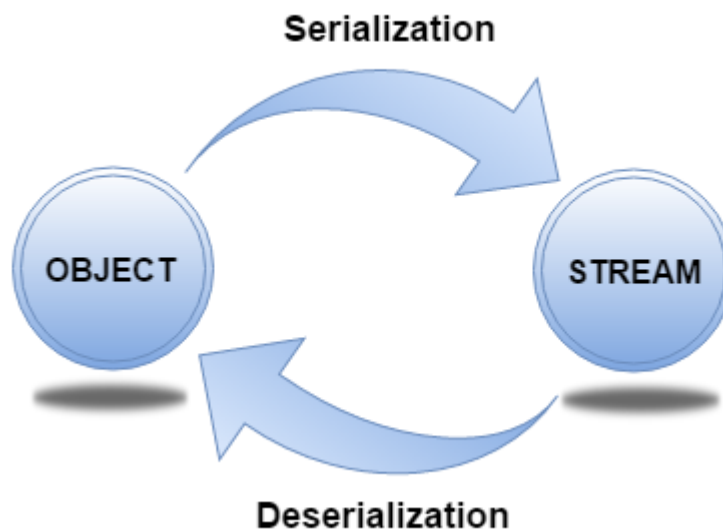
Answer.

Serialization in java is a mechanism of *writing the state of an object into a byte stream*.

It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called *deserialization*.

Advantage of Java Serialization

It is mainly used to travel object's state on the network (known as marshaling).

If a class that implements Serializable interface, then, its objects can be converted into stream and can travel through a network.