Q2.

1. .

**1.**throw**** keyword is used to throw a single exception explicitly from any method or constructor while ****throws**** keyword is used in method and constructor declaration, denoted which exception can possible be thrown by this method.

2.**throw** is followed by an instance of exception class while **throws** is followed by exception class name.

3**.throw** is used within the method and constructor where as **throws** is used with the method and constructor signature.

1. We can **throw** only single exceptions using throw but we can declare multiple exceptions using **throws** one of which may or may not throw by method.
2. .
3. Abstract class can have abstract and non-abstract methods.

Interface can have only abstract methods.It can have default and static methods also.

1. Abstract class doesn't support multiple inheritance.

Interface supports multiple inheritance.

1. Abstract class can have final, non-final, static and non-static variables.

Interface has only static and final variables.

1. Abstract class can provide the implementation of interface.

Interface can't provide the implementation of abstract class.

(c).

Assertion is achieved using the assert statement in Java. While executing assertion, it is believed to be true. If it fails, JVM throws an error named Assertion Error. It is mainly used for testing purposes during development.

Example:-

Int value = 15;

Assert value >= 20;

1. o.p(“Value :”+value);

Then it will throw an error ..

(d)

The try-with-resources statement is a try statement that declares one or more resources. The resource is as an object that must be closed after finishing the program. The try-with-resources statement ensures that each resource is closed at the end of the statement execution . In try-with-resources method there is no use of finally block. the file resource is opened in try block inside small brackets. Only the objects of those classes can be opened within the block which implements Auto Closeable interface and those object should also be local.

(e)

**Serialization** is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called **deserialization**.

We need serialization because the hard disk or network infrastructure are hardware component and we cannot send java objects because it understands just bytes and not java objects.