Java is an object-oriented language, everything revolve around the object. An object **represents runtime entity** of a class and is **essential to call variables and methods of the class**.

To create an object Java provides various ways that we are going to discuss in this topic.

1. New keyword
2. New instance
3. Clone method
4. Deserialization
5. NewInstance() method

1) new Keyword

In Java, creating objects using **new keyword** is very popular and common. Using this method user or system defined default constructor is called that initialize instance variables. And new keyword creates a memory area in heap to store created object.

**Example:**

In this example, we are creating an object by using new keyword.

public class NewKeyword

{

String s = "Let Us Learn";

public static void main(String as[])

{

NewKeyword a = new NewKeyword();

System.out.println(a.s);

}

}

2) New Instance

In this case, we use a static method forName() of **Class** class. This method loads the class and returns an object of type Class. That further we cast in our required type to get required type of object. This is what we do in this case.

**Example:**

You can see the example and understand that we loaded our class **NewInstance** by using **Class.forName()** method that returns the type Class object.

public class NewInstance

{

String a = "Let Us Learn";

public static void main(String[] args)

{

try

{

Class b = Class.forName("NewInstance");

NewInstance c = (NewInstance) b.newInstance();

System.out.println(c.a);

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

}

catch (InstantiationException e)

{

e.printStackTrace();

}

catch (IllegalAccessException e)

{

e.printStackTrace();

}

}

}

3) Clone() method

In Java, clone() is called on an object. When a clone() method is called JVM creates a new object and then copy all the content of the old object into it. When an object is created using the clone() method, a constructor is not invoked. To use the clone() method in a program the class implements the cloneable and then define the clone() method.

**Example:**

In this example, we are creating an object using clone() method.

public class CloneEg implements Cloneable

{

@Override

protected Object clone() throws CloneNotSupportedException

{

return super.clone();

}

String s = "Let Us Learn";

public static void main(String[] args)

{

CloneEg a= new CloneEg();

try

{

CloneEg b = (CloneEg) a.clone();

System.out.println(b.s);

}

catch (CloneNotSupportedException e)

{

e.printStackTrace();

}

}

}

4) deserialization

In Java, when an object is serialized and then deserialized, JVM create another separate object. When deserialization is performed JVM does not use any constructor for creating an object.

**Example:**

Lets see an example of creating object by deserialization concept.

import java.io.\*;

class DeserializationEg implements Serializable

{

private String a;

DeserializationEg(String name)

{

this.a = a;

}

public static void main(String[] args)

{

try

{

DeserializationEg b = new DeserializationEg("Let Us learn");

FileOutputStream c = new FileOutputStream("CoreJava.txt");

ObjectOutputStream d = new ObjectOutputStream(c);

d.writeObject(b);

d.close();

d.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

newInstance() method of Constructor class

In Java, Under **java.lang.reflect package** Constructor class is located. We can use it to create object. The Constructor class provides a method newInstance() that can be used for creating an object. This method is also **called a parameterized constructor.**

**Example:**

Lets create an example to create an object using newInstance() method of Constructor class.

import java.lang.reflect.\*;

public class ReflectionEg

{

private String s;

ReflectionEg()

{

}

public void setName(String s)

{

this.s = s;

}

public static void main(String[] args)

{

try

{

Constructorconstructor = ReflectionEg.class.getDeclaredConstructor();

ReflectionEg r = constructor.newInstance();

r.setName("Let Us Learn");

System.out.println(r.s);

}

catch (Exception e)

{

e.printStackTrace();

}

}

}