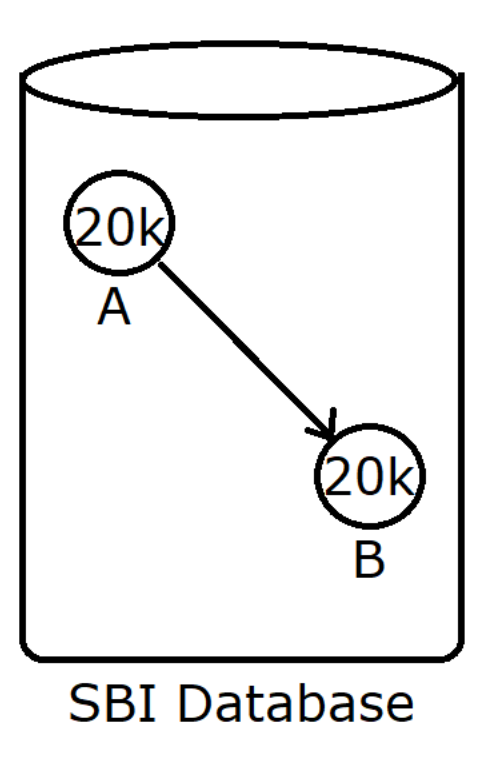
finally block : [100% Guarantee for Execution]

```
try
{
  database Connection code;
  Connection establishment code;
  Making transaction of 20K
  Updating Records          <= Exception Encounter
  Connection close;  X
  database close;  X
}
catch(Exception e)
{
   System.err.println("Exception Handled!!");
}
System.out.println("Program executed Normally");
```



Note : We should never write any closing statement in the **try block.**

* finally block is mainly used for Resource handling purpose. Our resourses are files, database, network resourses and so on, We need to close the resourses properly.

* Finally is a block which is **guaranteed for execution** regardless of exception that means whether the program contains exception or not, finally block will be excuted everytime.

* In only two cases the finally block will not be executed.
  a) If we write System.exit(0) in the try block as well as If this System.exit(0) statement is executed.
  b) If we write any infinite loop inside the try block then also finally block will not be executed.

* If we use the combination of try-finally (try and finally) then only resourses will be handled but not the execption, on the other hand, If we have a combination of try-catch-finally (try, catch and finally) then Exception and resourses both will be handled.

//Programs :

```
package com.ravi.basic;

public class FinallyBlock
{
    public static void main(String[] args)
    {
        System.out.println("Main method started");

        try
        {
            System.out.println(10/0);

        }
        finally
        {
            System.out.println("Guaranteed for Execution");
            System.out.println("Finally Block");
        }

        System.out.println("Main method ended");
    }

}
```

Note : In the above program we have abnormal termination.

```
package com.ravi.basic;

public class FinallyWithCatch
{
    public static void main(String[] args)
    {
        try
        {
            int []x = new int[-2];
            x[0] = 12;
            x[1] = 15;
            System.out.println(x[0]+" : "+x[1]);
        }
        catch(NegativeArraySizeException e)
        {
            System.err.println("Array Size is in negative value...");
        }
        finally
        {
            System.out.println("Resources will be handled here!!");
        }
        System.out.println("Main method ended!!!");
    }
}
```

In the above program we are handling the Exception and resourses both.

Limitation of finally block :

The following are the limitation of finally block :

1) In order to close the resourses, user is responsible to write finally block manually.

2) Due to finally block the length of the program will be increased.

3) In order to close the resourses inside the finally block, we need to declare the resourses outside of try block.[Otherwise Block level variable]

```
package com.ravi.basic;

import java.util.InputMismatchException;
import java.util.Scanner;

public class LimitationOfFinally
{
    public static void main(String[] args)
    {
        Scanner sc = null;
        try
        {
            sc = new Scanner(System.in);
            System.out.print("Enter Your Salary :");
            double sal = sc.nextDouble();
            System.out.println("Your Salary is :"+sal);
        }
        catch(InputMismatchException e)
        {
            System.out.println("Inside Catch");
            System.err.println("Input is Invalid!!!");
        }
        finally
        {
            System.out.println("Inside Finally");
            sc.close();
        }
    }
}
```

**try with resourses :
----

* try with resourses provides **automatic closing facility** of the resourses which is introduced from JDK 1.7V.

* In JDK 1.5V, Java software people has provided a predefiined interface called **Closeable** available in java.io package.

```
public interface Closeable
{
   void close() throws IOException ;
}
```

* All the Resourses classes which are implementing from Closeable interface, should override close method otherwise the corresponding class will become abstract class.
  Example :

```
public final class Scanner implmennts Closeable
{
    //Scanner class has to provide the support for close() method
}
```

* In JDK 1.7V, Java wanted to **provide automatic closing facilty** with the help of **try block** so, Java has introduced a new interface called **AutoCloseable as shown below**

```
public interface AutoCloseable
{
    void close() throws Exception;
}
```

* In order to get this auto closing facility by using AutoCloseable interface we need to perform the following two taks :
  a) The resourse class which we want to close automatically, that class must implements from AutoCloseable interface. [To provide close method support]
     Example :

```
public class DatabaseResourse implements AutoCloseable
{
  //We need to override close() method
}
```

  b) We need to pass the **Object reference of** AutoCloseable class (DatabaseResourse class) inside the try block as a parameter.
     Example :

```
try(DatabaseResourse dr = new DatabaseResourse())
{

}
```

  Note : Automatically try block (**try()**) will call close method of the given class i.e DatabaseResourse class.

* From JDK 1.7V, Closeable interface started extending AutoCloseable interface so, Closeable interface also provides Auto closing facility from JDK 1.7..

* Java compiler autaomatically insert **finally block**, whenver we use **try()** [try with resourses]

* From java 9V, We have an enhancement over try with resourses that we can write resourse classes outside of the try block, only reference is reqd.

* The follwoing 3 cases defines the writing style of try with resourses

Case 1:

```
try(resource1 ; resource2)  //Only the resources will be handled
{
}
```

Case 2 :

```
//Resources and Exception both will be  handled
try(resource1 ; resource2)
{
}
catch(Exception e)
{
}
```

Case 3 :
----
try with resourses enhancement from java 9v

```
DatabaseResourse d1 = new DatabaseResourse();
FileResourse f2 = new FileResourse();

try(d1; f2)
{
}
catch(Exception e)
{
}
```

Note : The DatabaseResourse class and FileResourse class must implements either from Closeable or AutoCloseable.

//Program :

```
package com.ravi.try_with_resourses;

import java.io.Closeable;
import java.io.IOException;

class DatabaseResourse implements AutoCloseable
{
    @Override
    public void close() throws Exception
    {
        System.out.println("Database Resourse is Closed");
    }
}

class FileResourse implements Closeable
{
    @Override
    public void close() throws IOException
    {
        System.out.println("File Resourse is Closed!!!");
    }
}

public class TryWithResoursesDemo1
{
    public static void main(String[] args) throws Exception
    {
        DatabaseResourse dr = new DatabaseResourse();
        FileResourse fr = new FileResourse();

        try(dr ; fr)
        {
            System.out.println("Try Block");
            System.out.println(10/0);

        }
        catch(ArithmeticException e)
        {
            System.out.println("Divide by zero Problem");
        }

    }
}
```
