```
/*  #CONCEPT1: We cannot use try alone
public class Ex1
{
   public static void main(String[] args)
   {
      try
      {


      }
   }
}
Output
Ex1.java:4: error: 'try' without 'catch', 'finally' or resource declarations
      try
      ^
1 error */



 /* #CONCEPT2: We cannot use catch() alone
public class Ex1
{
   public static void main(String[] args)
   {
      catch(Exception e)
      {
         System.out.println(e);
      }
   }
}
```

```
//Output
Ex1.java:22: error: 'catch' without 'try'
      catch(Exception e)
      ^
1 error*/



/* #CONCEPT3: We cannot use any statement between try and catch block
public class Ex1
{
   public static void main(String[] args)
   {
      System.out.println("1");
      int a=10,b=0,c;
      try
      {
         System.out.println("2");
         c=a/b;
         System.out.println("3");
         System.out.println(c);
      }
      System.out.println("4");
      catch(ArithmeticException e)
      {
         System.out.println("5");
         System.out.println(e);
         System.out.println("Division by zero is not possible");
      }
      System.out.println("6");
   }
}
```

Output

```
Ex1.java:44: error: 'try' without 'catch', 'finally' or resource declarations
      try
      ^
Ex1.java:52: error: 'catch' without 'try'
      catch(ArithmeticException e)
      ^
2 errors*/

/*#CONCEPT4: Control flow of try catch block
public class Ex1
{
   public static void main(String[] args)
   {
      System.out.println("1");
      int a=10,b=0,c;
      try
      {
         System.out.println("2");
         c=a/b;
         System.out.println("3");
         System.out.println(c);
      }
      catch(ArithmeticException e)
      {
         System.out.println("4");
         System.out.println(e);
         System.out.println("Division by zero is not possible");
      }
      System.out.println("5");
   }
```

```
}
Output
1
2
4
java.lang.ArithmeticException: / by zero
Division by zero is not possible
5 */



/*#CONCEPT5: in multiple catch block we cannot use Exception class in first catch block
and its child classes in succeding catch block
public class Ex1
{
   public static void main(String args[])
   {
      try
      {
         int n1,n2;
         n1=Integer.parseInt(args[0]);
         n2=Integer.parseInt(args[1]);
         System.out.println("Div result="+n1/n2);
      }
      catch(Exception e)
      {
         System.out.println("You cannot divide a number by zero");
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
      System.out.println("Please Enter two numbers");
```

```
    }
    catch(NumberFormatException e)
    {
    System.out.println("NumberFormatException generated");
    }
  }
  Output
  Ex1.java:119: error: exception ArrayIndexOutOfBoundsException has already been caught
    catch(ArrayIndexOutOfBoundsException e)
    ^
Ex1.java:123: error: exception NumberFormatException has already been caught
    catch(NumberFormatException e)
    ^
2 errors */


/*#CONCEPT6: in multiple catch block we can use child class of Exception class in first catch block
and its parent Exception class in succeding catch block
public class Ex1
{
  public static void main(String args[])
  {
    try
    {
      int n1,n2;
      n1=Integer.parseInt(args[0]);
      n2=Integer.parseInt(args[1]);
      System.out.println("Div result="+n1/n2);
    }
    catch(ArithmeticException e)
    {
```

```
            System.out.println("You cannot divide a number by zero");
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
        System.out.println("Please Enter two numbers");
        }
        catch(Exception e)
        {
        System.out.println("NumberFormatException generated");
        }
    }
}
Output
No Compiler Error */


/*#CONCEPT7: in multiple catch blocks we can use different child classes of Exception class
public class Ex1
{
    public static void main(String args[])
    {
        try
        {
            int n1,n2;
            n1=Integer.parseInt(args[0]);
            n2=Integer.parseInt(args[1]);
            System.out.println("Div result="+n1/n2);
        }
        catch(ArithmeticException e)
        {
            System.out.println("You cannot divide a number by zero");
```

```
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
        System.out.println("Please Enter two numbers");
        }
        catch(NumberFormatException e)
        {
        System.out.println("NumberFormatException generated");
        }
    }
}
Output
No compiler Error */



/*#CONCEPT8: in multiple catch blocks we cannot use same child classes of Exception class
public class Ex1
{
    public static void main(String args[])
    {
        try
        {
            int n1,n2;
            n1=Integer.parseInt(args[0]);
            n2=Integer.parseInt(args[1]);
            System.out.println("Div result="+n1/n2);
        }
        catch(ArithmeticException e)
        {
            System.out.println("You cannot divide a number by zero");
        }
```

```
        catch(ArithmeticException e)
        {
        System.out.println("Please Enter two numbers");
        }
        catch(ArithmeticException e)
        {
        System.out.println("NumberFormatException generated");
        }
    }
}
```

Output

```
PS D:\java\Java2_Programs\Unit3> javac Ex1.java
Ex1.java:218: error: exception ArithmeticException has already been caught
        catch(ArithmeticException e)
        ^
Ex1.java:222: error: exception ArithmeticException has already been caught
        catch(ArithmeticException e)
        ^
2 errors */
```

```
/*#CONCEPT9: The given below try-catch combination is possible.
try // Outer try block if exception occurs here then it will be caught  in outer catch block
{
    try // Inner try block if exception occurs here then it will be caught  in inner catch block
    {

    }
    catch()
    {
```

```
    }
}
catch() // Outer catch block
{

}
// Example1 of concept9
public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
            int a=10,b=0,c;
            System.out.println(a);
            try
            {
                c=a/b;
            }
            catch(ArithmeticException e)
            {
                System.out.println("Division by zero is not possible");


            }



        }
        catch(NullPointerException e)
        {
            System.out.println("NPE");
        }
```

```
        System.out.println("Hello");
    }
}
```

Output: (Note the control flow of execution)

```
10
Division by zero is not possible
Hello
```

```
// Example2 of Concept9
public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
            int a=10,b=0,c;
            System.out.println(a);
            String s= null;
            System.out.println(s.length());
            try
            {
                c=a/b;
            }
            catch(ArithmeticException e)
            {
                System.out.println("Division by zero is not possible");

            }


        }
```

```
        catch(NullPointerException e)
        {
           System.out.println("NPE");
        }
        System.out.println("Hello");
    }
}
Output(Note control flow of execution)
10
NPE
Hello */


/*#CONCEPT10: The given below try-catch combination is possible.
try  //if exception occurs here then it will caught in catch() block 1
{

}
catch() // catch() block1
{
    try //if exception occurs here then it will caught in catch() block 2
    {

    }
    catch() // catch() block2
    {

    }
}
public class Ex1
{
```

```
    public static void main(String[] args)
    {
        try
        {
        String s =null;
        System.out.println(s.length());
        }
        catch(NullPointerException e)
        {
            System.out.println("NPE");
            try
            {
                int a=10,b=2,c;
                System.out.println(a/b);
            }
            catch(ArithmeticException e1)
            {
                System.out.println("Division by zero is not possible");
            }
        }
        System.out.println("Hello");

    }
}
Output
NPE
5
Hello */


/* #CONCEPT11: We cannot use finally block alone
```

```
public class Ex1
{
   public static void main(String[] args)
   {
      finally
      {


      }
   }
}
```

Output

```
Ex1.java:177: error: 'finally' without 'try'
      finally
      ^
1 error */
```

```
/*#CONCEPT12: We can use finally block with try. if exception occurs then
 program will terminate abnormally and before that finally block will be
 executed.

public class Ex1
{
   public static void main(String[] args)
   {
      try
      {
         int a=10,b=0,c;
         System.out.println(a);
         c=a/b;
      }
```

```
        finally
        {

            System.out.println("This block will be executed always");
        }
        System.out.println("hello");
    }
}
```

Output

```
10
This block will be executed always
xception in thread "main" java.lang.ArithmeticException: / by zero
        at Ex1.main(Ex1.java:269) */
```

```
/*#CONCEPT13: We can use finally block with try. if exception does not occur then
 first try block anD then finally block will be executed. And if there are any
 statements after finally block then they will be executed.

public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
            int a=10,b=0,c;
            System.out.println(a);

        }
```

```
        finally
        {

            System.out.println("This block will be executed always");
        }
        System.out.println("hello");
    }
}
Output
10
This block will be executed always
hello */


/*#CONCEPT14: We cannot use catch() block after try finally block

public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
            int a=10,b=0,c;
            System.out.println(a);
            c=a/b;
        }
        finally
        {
            System.out.println("This block will be executed always");
        }
        catch(ArithmeticException e)
```

```
      {
         System.out.println("Division by zero is not possible");
      }

      System.out.println("hello");
   }
}
```

Output

```
Ex1.java:465: error: 'catch' without 'try'
      catch(ArithmeticException e)
      ^
1 error */
```

```
/*#CONCEPT15: We can use finally block with try catch. if exception occurs in try block then
 it will be caught in catch block And then finally block will be executed and then after rest
 of the statements (if any) will be executed.

public class Ex1
{
   public static void main(String[] args)
   {
      try
      {
         int a=10,b=0,c;
         System.out.println(a);
         c=a/b;

      }
      catch(ArithmeticException e)
      {
         System.out.println("Division by zero is not possible");
```

```
        }


        finally
        {


            System.out.println("This block will be executed always");
        }
        System.out.println("hello");
    }
}
```

Output

```
10
Division by zero is not possible
This block will be executed always
hello */
```

```
/*#CONCEPT16: We can use finally block with try catch. if exception does not occur in try block then
 execution control flow is from try to finally block (catch block will not be executed) and then after rest of
 the statements(if any).


 public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
            int a=10,b=0,c;
            System.out.println(a);
```

```
        }
        catch(ArithmeticException e)
        {
           System.out.println("Division by zero is not possible");
        }

        finally
        {

           System.out.println("This block will be executed always");
        }
        System.out.println("hello");
     }
}
```

Output

```
10
This block will be executed always
hello */



/*#CONCEPT17: in try finally block, inside finally we can use try catch() block*/


/* Example1 of CONCEPT17
 public class Ex1
{
    public static void main(String[] args)
    {
        try
        {
           int a=10,b=0,c;
```

```
        System.out.println(a);
      }
      finally
      {
        try
        {
          System.out.println(5/0);
        }
        catch(ArithmeticException e)
        {
          System.out.println("Division by zero is not possible");
        }
          System.out.println("This block will be executed always");
      }
      System.out.println("hello");
   }
}
```

Output

```
10
Division by zero is not possible
This block will be executed always
hello */

/*  Example2 of CONCEPT17
 public class Ex1
{
   public static void main(String[] args)
   {
      try
      {
        int a=10,b=0,c;
```

```
            System.out.println(a);
            System.out.println(a/b);
        }
        finally
        {
            try
            {
                System.out.println(5/0);
            }
            catch(ArithmeticException e)
            {
                System.out.println("Division by zero is not possible");
            }
                System.out.println("This block will be executed always");
        }
        System.out.println("hello");
    }
}
```

Output

```
10
Division by zero is not possible
This block will be executed always
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Ex1.main(Ex1.java:590) */
```

```
/* Program1: Write a program to handle NullPointerException use try catch block
public class Ex1
{
```

```
   public static void main(String[] args)
   {
      try
      {
      String s=null;
      System.out.println(s.length());
      }
      catch(NullPointerException e)
      {
         System.out.println("NullPointerException");
      }

   }
```

Output

```
PS D:\java\Java2_Programs\Unit3> java Ex1
NullPointerException */
```

/* Program2: Ask user to enter two integer values using command line argument.

WAP to handle ArrayIndexOutOfBoundsException,ArithmeticException and NumberFormatException

use multiple try catch block

```
public class Ex1
{
   public static void main(String args[])
   {
      try
      {
         int n1,n2;
         n1=Integer.parseInt(args[0]);
         n2=Integer.parseInt(args[1]);
```

```
        System.out.println("Div result="+n1/n2);
      }
      catch(ArithmeticException e)
      {
        System.out.println("You cannot divide a number by zero");
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
      System.out.println("Please Enter two numbers");
      }
      catch(NumberFormatException e)
      {
      System.out.println("NumberFormatException generated");
      }
   }
}
```

Output

```
PS D:\java\Java2_Programs\Unit3> java Ex1 6 2
Div result=3
PS D:\java\Java2_Programs\Unit3> java Ex1 6 0
You cannot divide a number by zero
PS D:\java\Java2_Programs\Unit3> java Ex1 6
Please Enter two numbers
PS D:\java\Java2_Programs\Unit3> java Ex1 6 L
NumberFormatException generated */
```

```
/* Program3: Ask user to enter two integer values using command line argument.
 WAP to handle ArrayIndexOutOfBoundsException,ArithmeticException and
NumberFormatException
 use  nested try and multiple try catch block.
```

```
import java.util.*;
 public class Ex1
{
   public static void main(String args[])
   {



      try
      {
         int n1,n2;
         n1=Integer.parseInt(args[0]);
         n2=Integer.parseInt(args[1]);
         if(n2==0)
         {
         try
         {
            System.out.println("Div result="+n1/n2);
         }
         catch(ArithmeticException e)
         {
            System.out.println("You cannot divide a number by zero");
         }
         }
         else
         {
            System.out.println("Div result="+n1/n2);
         }
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
```

```
        System.out.println("Please Enter two numbers");
        }
        catch(NumberFormatException e)
        {
        System.out.println("NumberFormatException generated");
        }
        System.out.println("hello");
    }
}
```

Output

```
PS D:\java\Java2_Programs\Unit3> java Ex1 8 2
Div result=4
PS D:\java\Java2_Programs\Unit3> java Ex1 8 0
You cannot divide a number by zero
PS D:\java\Java2_Programs\Unit3> java Ex1 8
Please Enter two numbers
PS D:\java\Java2_Programs\Unit3> java Ex1 8 u
NumberFormatException generated */
```