Define an Exception. Give an example.

Exception is an abnormal condition that terminates the program forcefully.

Example 1:

```
1   int div = 0;
2   div = 100/0;                        //abnormal condition
3   System.out.println(div);
```

Output:

```
java.lang.ArithmeticException : / by zero
```

Explanation: In given example 100 is divided by 0 which is mathematically not possible. So in java this statement will terminate the program by throwing an exception named ArithematicException.

Exception Handling

Exception Handling is a mechanism to handle abnormal condition so that the normal flow of application can be maintained. Exceptions usually disrupts the normal flow of the application that is why we need to handle them properly. Let's take a simple scenario:

Example 2:

```
1   statement 1;
2   statement 2;
3   statement 3;           //exception occurs
4   statement 4;
5   statement 5;
```

Suppose there are 5 statements in your program and there is an exception at statement 3, rest of the code (statement 4-5) won't execute. If we want our program to resume the normal flow then we need to perform exception handling.

Exception Handling in Java using try and catch blocks

The try-catch block is used to handle the exception. The code in which the exception may occur is enclosed in a try block, also called as a guarded region. The catch clause matches a specific exception to a block of code which handles that exception.

Example 3: Using try-catch to handle exceptions

```
1   int div = 0;
2   try
3   {
4       div=100/0;
5       System.out.println(div);
6   }
7   catch(ArithmeticException e)
```

```
8  {
9      System.out.println("Inside Catch: Division by zero not possible");
10 }
11 System.out.println("Resuming normal program flow");
```

Output:

```
Inside Catch: Division by zero not possible
Resuming normal program flow
```

Explanation: In the above program we have used a try block to cover a section of code that can cause an exception. At runtime line 4 will cause an exception to occur because of division by 0. Since we have used try-catch, the program control will automatically jump to the catch block. Post catch block's execution code outside catch block will run.

Working of try-catch block

- **try:** try blocks are used to write the statements where exception may occur. It must be used within the method.
- **catch:** catch blocks are used to handle the Exceptions. It must be used after the try block only.

Other Examples of Different types of exception classes

Example 4.1: ArrayIndexOutOfBoundsException

```
1  try
2  {
3      int arr[] = {4,9,10,2};
4      arr[7] = 10;
5  }
6  catch(ArrayIndexOutOfBoundsException e)    //Exception class
7  {
8      System.out.println("Inside Catch");
9      System.out.println(e);                 //Printing class name and message
10 }
```

Output:

```
Inside Catch
java.lang.ArrayIndexOutOfBoundsException: 7
```

Explanation: ArrayIndexOutOfBoundsException occurs when we try to access an index which is outside the range of given array. Range is from 0 to (array-length – 1). In above example we are accessing element at index 7 which is greater than array size of 4.

Example 4.2: StringIndexOutOfBoundsException

```
1  try
2  {
3      String str = "Hello";
4      char ch = str.charAt(-1);
5  }
6  catch(StringIndexOutOfBoundsException e)   //Exception class
7  {
```

```
8      System.out.println("Inside Catch");
9      System.out.println(e);                  //Printing class name and message
10 }
```

Output:

```
Inside Catch
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
```

Explanation: StringIndexOutOfBoundsException occurs when we try to access an index which is outside the range of given string. Range is from 0 to (string-length – 1). In above example we are accessing character at index -1 which is less than 0 (first index).

Example 4.3: NullPointerException

```
1  try
2  {
3     String str = null;
4     int index = str.indexOf('e');
5  }
6  catch(NullPointerException e) //Exception class
7  {
8     System.out.println("Inside Catch");
9     System.out.println(e);                  //Printing class name and message
10 }
```

Output:

```
Inside Catch
java.lang.NullPointerException
```

Explanation: NullPointerException occurs when we try to operate on null. In above example we are calling a function on null reference.

Problem 1	**Give output for each of the following code segments.** **Note: Explanation of each answer is mandatory.**

Code 1.1

```
1  try
2  {
3     int a = 0;
4     int b = 5 / a;
5  }
6  catch(ArithmeticException e)
7  {
8     System.out.println(e);
9  }
```

Code 1.2

```
1  int[] arr = new int[2];
2  try
3  {
```

```
4       arr[2] = arr[1] + 2;
5       System.out.println(arr[2]);
6   }
7   catch(ArrayIndexOutOfBoundsException e)
8   {
9       System.out.println(e);
10  }
```

Generic Exception

If your code throws an exception in try but catch is expecting different kind of exception then catch won't run. To create a generic catch that handles all kinds of exceptions you can use Exception class.

Example 5.1: try throws different exception than what is mentioned in catch

```
1   try
2   {
3       int arr[] = {4,9,10,2};
4       arr[7] = 10;
5   }
6   catch(NullPointerException e)              //Wrong type of exception class
7   {
8       System.out.println("Inside Catch");    //Will not run
9   }
```

Output:

```
java.lang.ArrayIndexOutOfBoundsException: 7
```

Explanation: ArrayIndexOutOfBoundsException occurs inside try however our catch is not equipped to handle it. In this case program execution will end because exception remained unhandled.

Example 5.2: Using Generic Exception

```
1   try
2   {
3       String str = "Hello";
4       char ch = str.charAt(-1);
5   }
6   catch(Exception ex)                        //Generic Exception class
7   {
8       System.out.println("Inside Catch");
9       System.out.println(ex);                //Printing class name and message
10  }
```

Output:

```
Inside Catch
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
```

Explanation: We have used generic Exception class in catch. Now this catch can handle any kind of exceptions thrown inside try.

Problem 2	Give output for each of the following code segments. Note: Explanation of each answer is mandatory.

Code 2.1

```
1  try
2  {
3     int a, b;
4     b = 0;
5     a = 5 / b;
6     System.out.print("A");
7  }
8  catch(Exception e)
9  {
10    System.out.print("B");
11 }
12 System.out.print("C");
```

Code 2.2

```
1  try
2  {
3     String str = "abc";
4     System.out.print("A");
5     char ch = str.charAt(3);
6  }
7  catch(ArrayIndexOutOfBoundsException e)
8  {
9     System.out.print("Index Out of Range");
10 }
```

Code 2.3

```
1  String[] st = new String[3];
2  try
3  {
4     st[2] = "abc";
5     String str = st[2];
6     System.out.print("A");
7     char ch = str.charAt(3);
8  }
9  catch(Exception e)
10 {
11    System.out.print("Index Out of Range");
12 }
```

Syntax Errors

Syntax errors cannot be handled using try catch blocks. A syntax error will generate a compile time error.

Empty Catch Block

You can leave the catch block without writing any actual code to handle the exception caught.

Running code inside try (after the point of exception)

Once the control jumps to the catch block it never returns to the try block.

Problem 3	**Give output for each of the following code segments.** **Note: Explanation of each answer is mandatory.**

Code 3.1

```
1  try
2  {
3     String str = 123;
4  }
5  catch(Exception e)
6  {
7     System.out.print("Exception Handled");
8  }
```

Code 3.2

```
1  int sum = 10;
2  try
3  {
4     for(int i = -1; i < 3; i++)
5     {
6           sum = (sum / i);
7     }
8  }
9  catch(ArithmeticException e)
10 {
11    System.out.print("0");
12 }
13 System.out.print(sum);
```

Code 3.3

```
1  try
2  {
3     int[] arr = {5,1,2,4};
4     int sum = 0;
5     for(int i = 0; i <= arr.length; i++)
6     {
7           sum = sum + arr[i];
```

```
8            System.out.print(arr[i]);
9      }
10 }
11 catch(Exception e)
12 {
13    System.out.print("Exception Occurred");
14 }
```

Problem 4	What is Exception Handling? Explain with example?

Problem 5	Refer the code given below and answer the questions that follows.
Code	

```
1  String myName = "Vishal";
2  int len = myName.length();
3  char ch = myName.charAt(len);
4  System.out.print(ch);
```

Problem 5.1	Modify the code above and handle the exception that can occur. Specify the exact exception.
Problem 5.2	Handle the exception in above code using Generic Exception.