6.a write a java program that describes exception handling mechanism

```
public class ExceptionHandlingDemo {
     // A method that performs division and might throw an ArithmeticException
    public static int divide(int numerator, int denominator) {
        return numerator / denominator;
    }
    public static void main(String[] args) {
        try {
            // Code that might throw an exception
            int result = divide(10, 0); // This will cause an ArithmeticException
            System.out.println("Result of division: " + result); // This line won't be
executed
        } catch (ArithmeticException e) {
            // Catch block to handle a specific exception type (ArithmeticException)
            System.out.println("Error: Cannot divide by zero!");
            System.out.println("Exception details: " + e.getMessage());
        } catch (Exception e) {
            // Catch block to handle any other unexpected exceptions
            System.out.println("An unexpected error occurred.");
            System.out.println("Exception details: " + e.getMessage());
        } finally {
            // The finally block always executes, regardless of whether an exception
occurred or not
            System.out.println("The 'try-catch-finally' block has finished
execution.");
        }

        System.out.println("Program continues after exception handling.");
    }
}
```

Outpot:

```
Error: Cannot divide by zero!
Exception details: / by zero
The 'try-catch-finally' block has finished execution.
Program continues after exception handling.
```

6. b. write a java program that illustrate multiple catch blocks

```
public class MultipleCatchExample {

    public static void main(String[] args) {
        try {
            // Scenario 1: ArithmeticException (division by zero)
            int result = 10 / 0;
            System.out.println("Result of division: " + result); // This line will not
be reached

            // Scenario 2: ArrayIndexOutOfBoundsException
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[5]); // This line will not be reached if
ArithmeticException occurs first
```

```
            // Scenario 3: NullPointerException
            String str = null;
            System.out.println(str.length()); // This line will not be reached if
previous exceptions occur
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: Cannot divide by
zero.");
            System.out.println("Error message: " + e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught an ArrayIndexOutOfBoundsException: Array index
is out of bounds.");
            System.out.println("Error message: " + e.getMessage());
        } catch (NullPointerException e) {
            System.out.println("Caught a NullPointerException: Attempted to access a
null object.");
            System.out.println("Error message: " + e.getMessage());
        } catch (Exception e) { // A general catch block for any other unexpected
exceptions
            System.out.println("Caught a general Exception: An unexpected error
occurred.");
            System.out.println("Error message: " + e.getMessage());
        } finally {
            System.out.println("Finally block executed. This always runs.");
        }
        System.out.println("Program continues after the try-catch-finally block.");
    }
}
```

Output:

Caught an ArithmeticException: Cannot divide by zero.
Error message: / by zero
Finally block executed. This always runs.
Program continues after the try-catch-finally block.

6.c write a java program for creation of java built-in exceptions

```
public class BuiltInExceptionDemo {

  public static void main(String[] args) {
    // 1. ArithmeticException: Division by zero
    try {
      int result = 10 / 0;
      System.out.println("Result of division: " + result); // This line will not be executed
    } catch (ArithmeticException e) {
      System.err.println("ArithmeticException caught: " + e.getMessage());
    }

    // 2. ArrayIndexOutOfBoundsException: Accessing an invalid array index
    try {
      int[] numbers = {1, 2, 3};
```

```
            System.out.println("Element at index 3: " + numbers[3]); // This will throw an exception
        } catch (ArrayIndexOutOfBoundsException e) {
            System.err.println("ArrayIndexOutOfBoundsException caught: " + e.getMessage());
        }

        // 3. NullPointerException: Accessing a method or field of a null object
        try {
            String str = null;
            System.out.println("Length of string: " + str.length()); // This will throw an exception
        } catch (NullPointerException e) {
            System.err.println("NullPointerException caught: " + e.getMessage());
        }

        // 4. NumberFormatException: Attempting to convert an invalid string to a number
        try {
            String invalidNumber = "abc";
            int num = Integer.parseInt(invalidNumber);
            System.out.println("Parsed number: " + num); // This line will not be executed
        } catch (NumberFormatException e) {
            System.err.println("NumberFormatException caught: " + e.getMessage());
        }

        // 5. ClassCastException: Attempting to cast an object to an incompatible type
        try {
            Object obj = "Hello";
            Integer i = (Integer) obj; // This will throw an exception
            System.out.println("Casted integer: " + i); // This line will not be executed
        } catch (ClassCastException e) {
            System.err.println("ClassCastException caught: " + e.getMessage());
        }
    }
}
```

Output:

```
ArithmeticException caught: / by zero
ArrayIndexOutOfBoundsException caught: Index 3 out of bounds for length 3
NullPointerException caught: Cannot invoke "String.length()" because "str" is null
NumberFormatException caught: For input string: "abc"
ClassCastException caught: class java.lang.String cannot be cast to class java.lang.Integer
```

6.d write a java program for creation of User-defined exceptions

```
// Step 1: Create a user-defined exception class
class InvalidAgeException extends Exception {
   // Constructor with message
   public InvalidAgeException(String message) {
      super(message);
   }
}

public class UserDefinedExceptionDemo {

   // Step 2: A method that throws the custom exception
   public static void validateAge(int age) throws InvalidAgeException {
      if (age < 18) {
         throw new InvalidAgeException("Age must be 18 or above to vote.");
      } else {
         System.out.println("Welcome! You are eligible to vote.");
      }
   }

   public static void main(String[] args) {
      try {
         // Test case 1: invalid age
         validateAge(15);

         // Test case 2: valid age
         validateAge(25);
      } catch (InvalidAgeException e) {
         System.out.println("Caught Exception: " + e.getMessage());
      } finally {
         System.out.println("Validation process completed.");
      }
   }
}
```

Output:

```
Caught Exception: Age must be 18 or above to vote.
Validation process completed.
```