by Kunal Sir

Assignment: Exception Handling (finally Block)

Part A: MCQs

1. What is the main purpose of the `finally` block in Java?

A. To throw exceptions
B. To handle exceptions
C. To execute important code compulsorily
D. To stop program execution

Answer: C

---

2. When does the `finally` block execute?

A. Only when an exception occurs
B. Only when no exception occurs
C. Always, whether an exception occurs or not
D. Only if catch block is present

Answer: C

---

3. Which of the following is true about the `finally` block?

A. It executes before try block
B. It executes only if catch executes
C. It executes after try or catch block
D. It executes only during runtime error

Answer: C

---

4. Which block is mandatory to use the `finally` block?

A. catch
B. throw
C. try
D. throws

Answer: C

---

5. Can we use `finally` without a `catch` block?

A. Yes
B. No
C. Only in runtime exception
D. Only in checked exception

Answer: A

---

6. Which of the following is a common use of the `finally` block?

A. Reading user input
B. Closing database or file resources
C. Printing output
D. Throwing exceptions

Answer: B

---

7. What happens if an exception occurs inside the `try` block?

A. finally block is skipped
B. Program stops immediately
C. catch block executes first, then finally
D. finally executes before catch

Answer: C

---

8. Will the `finally` block execute if a `return` statement is present in the try block?

A. No
B. Yes
C. Only sometimes
D. Only if no exception occurs

Answer: B

---

9. In which case does the `finally` block NOT execute?

A. When exception occurs
B. When no exception occurs
C. When `System.exit(0)` is called
D. When catch block handles exception

Answer: C

---

10. If an exception occurs inside the `finally` block, what happens?

A. It is ignored
B. It behaves like any other exception
C. Program compiles but does not run
D. JVM ignores it

Answer: B

Part B: Problem Statements

Problem 1: Finally Block Execution (No Exception)

Problem Statement

Write a Java program that divides two numbers where no exception occurs.
Use a `finally` block to print a message showing that it executes even when the program runs normally.

Sample Input

```
Enter first number: 10
Enter second number: 2
```

Sample Output

```
Result: 5
Finally block executed
```

```java
class Problem1 {
    public static void main(String[] args) {
        int a = 10, b = 2;
        try {
            int result = a / b;
            System.out.println("Result: " + result);
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
```

Problem 2: Finally Block Execution (With Exception)

Problem Statement

Write a Java program that performs division by zero inside a `try` block.
Handle the exception using a `catch` block and use a `finally` block to print a compulsory message.

Sample Input

```
Enter first number: 10
Enter second number: 0
```

Sample Output

```
Error: Division by zero
Finally block executed
```

```java
class Problem2 {
    public static void main(String[] args) {
        int a = 10, b = 0;
        try {
            int result = a / b;
            System.out.println(result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero");
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
```

Problem 3: Try, Catch and Finally Order

Problem Statement

Write a Java program that prints messages inside the `try`, `catch`, and `finally` blocks. Cause an exception to observe the execution order.

Sample Output

```
Inside try block
Inside catch block
Inside finally block
```

```java
class Problem3 {
    public static void main(String[] args) {
        try {
            System.out.println("Inside try block");
            int a = 10 / 0;
        } catch (Exception e) {
            System.out.println("Inside catch block");
        } finally {
            System.out.println("Inside finally block");
        }
    }
}
```

Problem 4: Try and Finally Without Catch

Problem Statement

Write a Java program using only `try` and `finally` blocks.
The program should show that the `finally` block executes even without a `catch` block.

Sample Output

```
Inside try block
Inside finally block
```

```java
class Problem4 {
    public static void main(String[] args) {
        try {
            System.out.println("Inside try block");
        } finally {
            System.out.println("Inside finally block");
        }
    }
}
```

Problem 5: Finally Block with Return Statement

Problem Statement

Write a Java program where a `return` statement is written inside the `try` block.
Use the `finally` block to print a message after the return.

Sample Output

```
Inside try block
Inside finally block
```

```java
class Problem5 {
    static int demo() {
        try {
            System.out.println("Inside try block");
            return 10;
        } finally {
            System.out.println("Inside finally block");
        }
    }

    public static void main(String[] args) {
        demo();
    }
}
```

Problem 6: Simple Calculator Using Finally

Problem Statement

Create a Java program that performs division using user input.
Use a `finally` block to print **"Thank you for using the program"**.

Sample Input

```
Enter first number: 20
Enter second number: 4
```

Sample Output

```
Result: 5
Thank you for using the program
```

```java
class Problem6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Enter first number: ");
            int a = sc.nextInt();
            System.out.print("Enter second number: ");
            int b = sc.nextInt();
            int result = a / b;
            System.out.println("Result: " + result);
        } catch (Exception e) {
            System.out.println("Invalid input or division by zero");
        } finally {
            System.out.println("Thank you for using the program");
        }
    }
}
```

Problem 7: Finally Block with No Exception Handling

Problem Statement

Write a Java program that does not generate any exception but still uses `try` and `finally` blocks to print messages.

Sample Output

```
Program running normally
Finally block executed
```

```java
class Problem7 {
    public static void main(String[] args) {
        try {
            System.out.println("Program running normally");
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
```

Problem 8: Multiple Statements in Finally Block

Problem Statement

Write a Java program where the `finally` block contains multiple print statements. Show that all statements execute.

Sample Output

```
Inside try block
Cleaning resources
Closing program
Finally block completed
```

```java
class Problem8 {
    public static void main(String[] args) {
        try {
            System.out.println("Inside try block");
        } finally {
            System.out.println("Cleaning resources");
            System.out.println("Closing program");
            System.out.println("Finally block completed");
        }
    }
}
```

Problem 9: Exception Without Catch

Problem Statement

Write a Java program where an exception occurs in the `try` block but no `catch` block is used. Use a `finally` block to print a message before program termination.

Sample Output

```
Inside finally block
Exception in thread "main" java.lang.ArithmeticException
```

```java
class Problem9 {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } finally {
            System.out.println("Inside finally block");
        }
    }
}
```

Problem 10: Importance of Finally Block

Problem Statement

Write a Java program with comments explaining why the `finally` block is important. Use simple print statements to show its compulsory execution.

Sample Output

```
Program started
Finally block is important
Program ended
```

```java
class Problem10 {
    public static void main(String[] args) {
        System.out.println("Program started");
        try {
            int a = 10 / 2;
        } finally {
            System.out.println("Finally block is important");
        }
        System.out.println("Program ended");
    }
}
```
