Institute of Computer Technology

B. Tech. Computer Science and Engineering

Semester: III

Sub: Object-Oriented Programming

Course Code: 2CSE303

Practical Number: 11

Objective:

To learn about exception handling concepts in Java.

Q1. Explain exception, and exception handling concept in java.

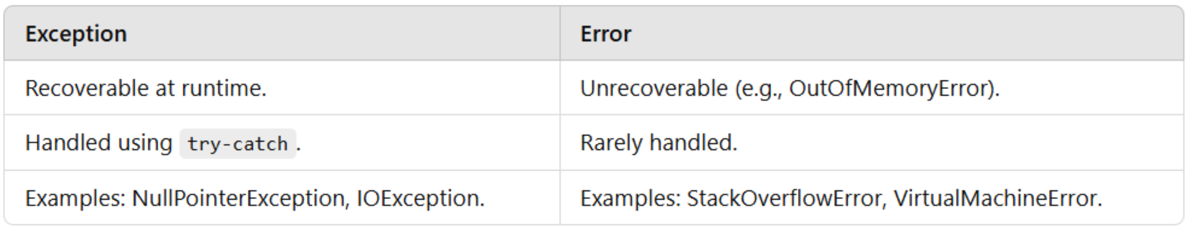
ANS :

Exception: An exception is an event that disrupts the normal flow of a program during runtime. It is an object representing an error condition.

Exception Handling: It is a mechanism to handle runtime errors, ensuring the normal flow of the program.

Q2. What is the difference between exception and error in java? Explain with an example.

ANS :



Exception	Error
Recoverable at runtime.	Unrecoverable (e.g., OutOfMemoryError).
Handled using `try-catch`.	Rarely handled.
Examples: NullPointerException, IOException.	Examples: StackOverflowError, VirtualMachineError.

Q3. Explain the concept of try and catch block in java exception handling with an appropriate program example.

Code :

```
public class TryCatchExample {
   public static void main(String[] args) {
      try {
         int data = 50 / 0;
      } catch (ArithmeticException e) {
         System.out.println("Caught Exception: " + e);
      }
   }
}
```

Q4. Explain throw and throws concept in java exception handling with an appropriate program example.

ANS :

Throw: Used to explicitly throw an exception.

Throws: Used in method declaration to indicate exceptions the method can throw.

E.g

```
class ThrowThrowsExample {
   void checkAge(int age) throws IllegalArgumentException {
      if (age < 18) {
         throw new IllegalArgumentException("Age is not valid!");
      }
   }
   public static void main(String[] args) {
      try {
         new ThrowThrowsExample().checkAge(15);
      } catch (IllegalArgumentException e) {
         System.out.println("Exception: " + e.getMessage());
      }
   }
}
```

Q5. Explain the concept of multiple catch block in java exception handling with an appropriate program example.

Code :

```
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            int[] arr = new int[5];
            arr[5] = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index Out of Bounds");
        }
    }
}
```

Q6. Explain the concept of finally block in java exception handling with an example.

Code :

```
public class FinallyExample {
    public static void main(String[] args) {
        try {
            int data = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Exception: " + e.getMessage());
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
```

Error :

OutOfMemoryError

Q7. Explain hierarchy of the exception class with an appropriate diagram example.

Code :

Output :

Q8. Write an appropriate program of the following checked exception

1. IOException.

Code :
```
import java.io.*;

public class IOExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("nonexistent.txt");
        } catch (IOException e) {
            System.out.println("IOException occurred: " + e.getMessage());
        }
    }
}
```

2. FileNotFoundException.

Code :
```
import java.io.*;
public class FileNotFoundExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("missingfile.txt");
        } catch (FileNotFoundException e) {
            System.out.println("FileNotFoundException occurred: " + e.getMessage());
        }
    }
}
```

3. <u>ClassNotFoundException.</u>

Code : public class ClassNotFoundExceptionExample {

```
    public static void main(String[] args) {
        try {
            Class.forName("NonExistentClass");
        } catch (ClassNotFoundException e) {
            System.out.println("ClassNotFoundException occurred: " + e.getMessage());
        }
    }
}
```

4. <u>SQLException.</u>

Code : import java.sql.*;

```
public class SQLExceptionExample {
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("invalid-url", "user", "pass");
        } catch (SQLException e) {
            System.out.println("SQLException occurred: " + e.getMessage());
        }
    }
}
```

5. <u>InterruptedException</u>

Code :
```
public class InterruptedExceptionExample {
    public static void main(String[] args) {
        Thread t = new Thread(() -> {});
        t.start();
        try {
            t.join();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException occurred: " + e.getMessage());
        }
    }
}
```

6. <u>Instantiation Exception</u>

Code :
```
public class InstantiationExceptionExample {
    public static void main(String[] args) {
        try {
            Class<?> clazz = Class.forName("java.util.ArrayList");
            Object obj = clazz.newInstance();
        } catch (InstantiationException | IllegalAccessException | ClassNotFoundException e) {
            System.out.println("Exception occurred: " + e.getMessage());
        }
    }
}
```