ADVENTIST UNIVERSITY OF CENTRAL AFRICA

Id: 25969

Names: Duhimbazimana Johnson

1. IOException (Read a Non-Existent File)

```
import java.io.*;
import java.util.Scanner;

public class IOExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the file name to open: ");
        String fileName = scanner.nextLine();

        try {
            // Attempting to open a file that may not exist
            FileInputStream fileInputStream = new FileInputStream(fileName);
            int data;
            while ((data = fileInputStream.read()) != -1) {
                System.out.print((char) data);
            }
            fileInputStream.close();
        } catch (IOException e) {
            // Handling IOException if the file does not exist
            System.out.println("Error: Unable to open the file - " + e.getMessage());
        }
    }
}
```

// Comment about this program: IOException occurs when there is an issue with input/output operations, such as accessing a file that doesn't exist.

2. FileNotFoundException (Delete a File)

```
import java.io.*;
import java.util.Scanner;

public class FileNotFoundExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the file name to delete: ");
        String fileName = scanner.nextLine();

        File file = new File(fileName);
        if (!file.exists()) {
            // File does not exist
            System.out.println("Error: File not found.");
        } else {
            // Attempting to delete the file
            if (file.delete()) {
                System.out.println("File deleted successfully.");
            } else {
                System.out.println("Error: File could not be deleted.");
            }
        }
    }
```

```
}
```

// Comment about this program: FileNotFoundException is a subclass of IOException that occurs specifically when a file-related operation is performed on a non-existent file.

3. EOFException (Read Beyond File Content)

```
import java.io.*;

public class EOFExceptionExample {
    public static void main(String[] args) {
        String fileName = "data.bin";

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fileName))) {
            // Writing data to a file
            oos.writeObject("Hello");
            oos.writeObject("World");
        } catch (IOException e) {
            System.out.println("Error: Unable to write data.");
        }

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {
            // Reading data until end of file is reached
            while (true) {
                System.out.println(ois.readObject());
            }
        } catch (EOFException e) {
            // End of file reached
            System.out.println("Reached end of file.");
        } catch (ClassNotFoundException | IOException e) {
```

```
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

// Comment about this program: EOFException occurs when attempting to read beyond the end of a file in a stream.

4. SQLException (Invalid Database Connection)

```
import java.sql.*;
import java.util.Scanner;

public class SQLExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter database URL: ");
        String dbUrl = scanner.nextLine();
        System.out.print("Enter username: ");
        String username = scanner.nextLine();
        System.out.print("Enter password: ");
        String password = scanner.nextLine();

        try (Connection conn = DriverManager.getConnection(dbUrl, username, password)) {
            // Attempting database connection
            System.out.println("Database connection successful!");
        } catch (SQLException e) {
            // Handling SQLException for invalid connection details
            System.out.println("Error: Could not establish a database connection.");
        }
    }
}
```

// Comment about this program: SQLException occurs during database operations, such as incorrect credentials or invalid connection strings.

5. ClassNotFoundException (Load a Missing Class)

```
import java.util.Scanner;

public class ClassNotFoundExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the class name to load: ");
        String className = scanner.nextLine();

        try {
            // Attempting to load a class by name
            Class<?> clazz = Class.forName(className);
            System.out.println("Class loaded: " + clazz.getName());
        } catch (ClassNotFoundException e) {
            // Handling ClassNotFoundException
            System.out.println("Class not found: " + e.getMessage());
        }
    }
}
```

// Comment about this program: ClassNotFoundException occurs when trying to load a class that is not available in the classpath.

6. ArithmeticException (Divide by Zero)

```
import java.util.Scanner;


public class ArithmeticExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        int numerator = scanner.nextInt();
        System.out.print("Enter the denominator: ");
        int denominator = scanner.nextInt();


        try {
            // Attempting division
            int result = numerator / denominator;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            // Handling division by zero
            System.out.println("Error: Division by zero is not allowed.");
        }
    }
}
```

// Comment about this program: ArithmeticException occurs when performing invalid arithmetic operations, such as dividing by zero.

7. NullPointerException (Access a Null Object)

```
public class NullPointerExceptionExample {
    public static void main(String[] args) {
        String[] data = null;

        try {
            // Attempting to access a property of a null object
            System.out.println("Data length: " + data.length);
        } catch (NullPointerException e) {
            System.out.println("Error: Attempt to access a null reference.");
        }
    }
}
```

// Comment about this program: NullPointerException occurs when trying to use an object that is null.

8**. ArrayIndexOutOfBoundsException (Access Invalid Index)**

```
public class ArrayIndexOutOfBoundsExceptionExample {
    public static void main(String[] args) {
        int[] array = {5, 10, 15, 20};

        try {
            // Accessing an invalid array index
            System.out.println("Array element at index 5: " + array[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index out of bounds.");
        }
    }
}
```

// Comment about this program: ArrayIndexOutOfBoundsException occurs when accessing an invalid index in an array.

9. ClassCastException (Invalid Casting)

```
public class ClassCastExceptionExample {
    public static void main(String[] args) {
        Object obj = new String("Java");

        try {
            // Attempting invalid casting
            Integer num = (Integer) obj;
        } catch (ClassCastException e) {
            System.out.println("Error: Cannot cast String to Integer.");
        }
    }
}
```

// Comment about this program: ClassCastException occurs when attempting to cast an object to a subclass it is not an instance of.

10. IllegalArgumentException (Invalid Argument)

```
public class IllegalArgumentExceptionExample {
    public static void main(String[] args) {
        try {
            // Providing invalid argument to a method
            Thread.sleep(-500);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: Invalid sleep time provided.");
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}
```

// Comment about this program: IllegalArgumentException occurs when an invalid argument is passed to a method.

11. NumberFormatException (Invalid String to Number)

```
import java.util.Scanner;

public class NumberFormatExceptionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        String input = scanner.nextLine();

        try {
            // Parsing a string to a number
            double number = Double.parseDouble(input);
            System.out.println("Parsed number: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid input, not a number.");
        }
    }
}
```

// Comment about this program: NumberFormatException occurs when attempting to parse an invalid string as a number.