

LAB # 13

Task No 01: Write a Fraction class that has a constructor that takes a numerator and a denominator. If the user passes in a denominator of 0, throw an exception of type `std::runtime_error` (included in the `stdexcept` header). In your main program, ask the user to enter two integers. If the Fraction is valid, print the fraction. If the Fraction is invalid, catch a `std::exception`, and tell the user that they entered an invalid fraction.

Code:**Main:**

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter the numerator: ");
            int numerator = scanner.nextInt();

            System.out.print("Enter the denominator: ");
            int denominator = scanner.nextInt();

            Fraction fraction = new Fraction(numerator, denominator);
            fraction.printFraction();
        } catch (InvalidFractionException e) {
            System.out.println("Invalid fraction: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Exception:

```
import java.util.Scanner;

class InvalidFractionException extends RuntimeException {
    public InvalidFractionException(String message) {
        super(message);
    }
}
```

Fraction:

```
class Fraction {
    private int numerator;
    private int denominator;

    public Fraction(int numerator, int denominator) {
        if (denominator == 0) {
            throw new InvalidFractionException("Invalid fraction: Division by zero");
        }
        this.numerator = numerator;
        this.denominator = denominator;
    }

    public void printFraction() {
        System.out.println(numerator + "/" + denominator);
    }
}
```

}

Output:

```
Enter the numerator: 3
Enter the denominator: 0
Invalid fraction: Invalid fraction: Division by zero
```

Task No 02: Write a program which implements Banking System by having all standard functionalities and will be implemented by branches. Try to identify and implement user defined exceptions for the system.

Hint:

- Create Bank Class
- public void Create Account(){}
 - public void deposit() throws Exception{
 - System.out.println("Enter Amount to be deposited:);
 - If(deposit>100000)
 - throw new Exception("\n you cant deposit this big amount");
 - Else
 - balance=balance+deposit;
- Public void withdraw() throws Exception{} (same logic as deposit)

Code:

Main:

```
public class BankingSystem {
    public static void main(String[] args) {
        Bank bank = new Bank();

        try {
            bank.createAccount();
            bank.deposit();
            bank.withdraw();
        } catch (LimitExceededException e) {
            System.out.println("Exception occurred: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Exception:

```
import java.util.Scanner;
class LimitExceededException extends Exception {
    public LimitExceededException(String message) {
        super(message);
    }
}
```

Bank:

```
class Bank {
    private double balance;
```

```
public Bank() {
    balance = 0;
}
public void createAccount() {
    System.out.println("Account created successfully!");
}
public void deposit() throws LimitExceededException {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter amount to be deposited: ");
    double deposit = scanner.nextDouble();
    scanner.nextLine();

    if (deposit > 100000) {
        throw new LimitExceededException("\nYou can't deposit this big amount");
    } else {
        balance += deposit;
        System.out.println("Deposit successful!");
    }
}
public void withdraw() throws LimitExceededException {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter amount to be withdrawn: ");
    double withdrawal = scanner.nextDouble();
    scanner.nextLine();

    if (withdrawal > 100000) {
        throw new LimitExceededException("\nYou can't withdraw this big amount");
    } else if (withdrawal > balance) {
        throw new LimitExceededException("\nInsufficient balance");
    } else {
        balance -= withdrawal;
        System.out.println("Withdrawal successful!");
    }
}
}
```

Output:

```
Account created successfully!
Enter amount to be deposited: 200000
Exception occurred:
You can't deposit this big amount
```