Task # 1: 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.

Solution:

FRACTION CLASS

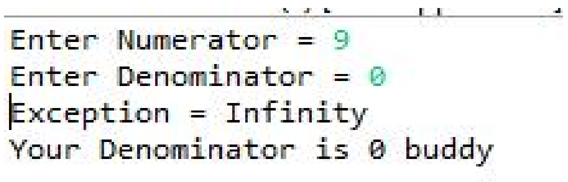
```
package task1;
public class Fraction {
	private int numerator;
	private int denominator;
	public Fraction(int numerator, int denominator) {
		this.numerator = numerator;
		this.denominator = denominator;
	}
	@Override
	public String toString() {
		return "Fraction number = " + numerator/denominator;
	}
}
```

MAIN METHOD

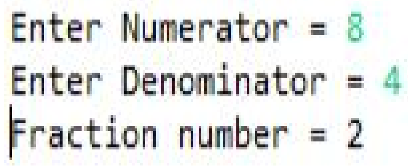
```
package task1;
import java.util.Scanner;
public class Main {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		  int Numerator , denominator;
		 try
		 {
		    System.out.print("Enter Numerator = ");
		    Numerator  = input.nextInt( );
		    System.out.print("Enter Denominator = ");
		    denominator = input.nextInt( );
		    Fraction f1 = new Fraction(Numerator,denominator);
		    if (denominator < 1)
		      throw new Exception("Exception = Infinity");
		    System.out.println(f1.toString());
		 }
		 catch(Exception e)
```

```
        {
            System.out.println(e.getMessage( ));
            System.out.println("Your Denominator is 0 buddy");
        }
    }
}
```

OUTPUT:



```
Enter Numerator = 9
Enter Denominator = 0
Exception = Infinity
Your Denominator is 0 buddy
```



```
Enter Numerator = 8
Enter Denominator = 4
Fraction number = 2
```

Task # 2: 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)

Solution

INTERFACE BANKING SYSTEM

```
public interface Banking_System {
    void CreateAccount ();
    void Search_Account();
   void details ();
    void Update_CustInfo();
    void Cash_Withdraw();
   void Cash_Deosit();}
```

BANK

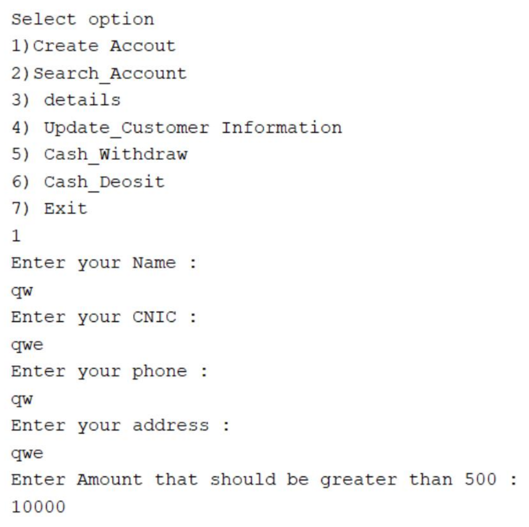
```
public class Bank implements Banking_System {
    String name, CNIC, Phone, address;
    int cash, withdraw, deposit;
    Scanner s = new Scanner(System.in);
    public void CreateAccount() {
        try {
            System.out.println("Enter your Name :");
            name = s.nextLine();
            System.out.println("Enter your CNIC :");
            CNIC = s.nextLine();
            System.out.println("Enter your phone :");
            Phone = s.nextLine();
            System.out.println("Enter your address :");
            address = s.nextLine();
            System.out.println("Enter Amount that should be greater than 500 :");
            cash = s.nextInt();
            if (cash < 500) {                 throw new Exception("Exception:Errorrrrrrrrr........");       }
        } catch (Exception e) {           System.out.println("insufficient Amount");       }              }
        void printstate() {
        System.out.println("======BANK======");
        System.out.println("Your name is       :" + name);
        System.out.println("Your CNIC is         :" + CNIC);
        System.out.println("Your phone is      :" + Phone);
        System.out.println("Your address is     :" + address);
        System.out.println("Your total cash is :" + cash);     }
        public void Search_Account() {       printstate();          }
        public void details() {                        printstate();          }
       public void Update_CustInfo() {     CreateAccount();    }
public void Search_Account() {            printstate();           }
   public void details() {                         printstate();           }
   public void Update_CustInfo() {CreateAccount();   }
   public void Cash_Withdraw() {
      try {
       System.out.println("Enter Amount to Withdraw");
          withdraw = s.nextInt();
          if (withdraw > 50000) {
            throw new Exception("\n ====!you cant withdraw this big amount");
          } else {               cash = cash - withdraw;     }
         System.out.println("Your withdraw amount: " + withdraw);
         System.out.println("Your remaining cash is:" + cash);
      } catch (Exception e) {
         System.out.println(e.getMessage());
         System.out.println("insufficient Amount");      }  }
   public void Cash_Deosit() {
      try {
         System.out.println("Enter Amount to deposit");
         deposit = s.nextInt();
         if (deposit > 100000) {
            throw new Exception("\n you cant deposit this big amount");
         } else {             cash += deposit;      }
         System.out.println("Your deposit amount: " + deposit);
         System.out.println("Your updated cash is:" + cash);
      } catch (Exception e) {
         System.out.println("insufficient Amount");      }   }
```

MAIN METHOD

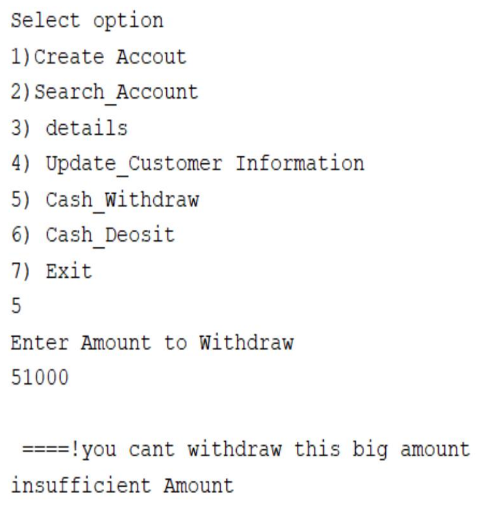
```
public static void main(String[] args) {
      String w;
      Scanner e = new Scanner(System.in);
      do {
         Bank a = new Bank();
         System.out.println("Select option");
         System.out.println("1)Create Accout");
         System.out.println("2)Search_Account");
         System.out.println("3) details");
         System.out.println("4) Update_Customer Information");
         System.out.println("5) Cash_Withdraw");
         System.out.println("6) Cash_Deosit");
         System.out.println("7) Exit");
```

```
        char d = e.next().charAt(0);
        switch (d) {
            case '1':           a.CreateAccount();          break;
            case '2':           a.Search_Account();         break;
            case '3':           a.details();                break;
            case '4':           a.Update_CustInfo();        break;
            case '5':           a.Cash_Withdraw();          break;
            case '6':           a.Cash_Deosit();            break;
            case '7':           break;
            default:            break; }
    } while (true); }
```

Output



```
Select option
1)Create Accout
2)Search_Account
3) details
4) Update_Customer Information
5) Cash_Withdraw
6) Cash_Deosit
7) Exit
1
Enter your Name :
qw
Enter your CNIC :
qwe
Enter your phone :
qw
Enter your address :
qwe
Enter Amount that should be greater than 500 :
10000
```



```
Select option
1)Create Accout
2)Search_Account
3) details
4) Update_Customer Information
5) Cash_Withdraw
6) Cash_Deosit
7) Exit
5
Enter Amount to Withdraw
51000

 ====!you cant withdraw this big amount
insufficient Amount
```