TASK:

Create BankAccount class which has private fields accountNumber, balance

Add constructor to initialize the accountNumber and balance.

Create methods deposit() and withdraw(), where :

Deposit() adds to the balance.

Withdraw() subtracts from the balance, but only if sufficient funds are available.

In the main method, create an object of BankAccount, and demonstrate deposit and withdraw operations.

package bankAccountQuestion;  
  
public class BankAccount {  
 //Private fields to store account details  
 private int accountNumber;  
 private double balance;  
 //=============================================================================  
  
 //Constructor to initialize accountNumber and balance  
 public BankAccount(int accountNumber, double balance){  
 this.accountNumber=accountNumber;  
 this.balance=balance;  
 }  
 //=============================================================================  
  
 //Method to deposit money into account  
 public void deposit(double amount){  
 if (amount > 0){ //check if amount is positive  
 balance += amount; // Adds the amount to the balance to  
 System.*out*.println("Deposited: "+ amount + ", New Balance: "+ balance);  
 }else {  
 System.*out*.println("Deposit Amount should be positive");  
 }  
 }  
  
 //=============================================================================  
  
 //Method to withdraw money  
 public void withdraw(double amount){  
 if (amount > 0 ){ //Check if the withdrawal amount is positive  
 if (balance >= amount){ // checking if sufficient funds are available  
 balance -= amount; //deducting the amount from the balance  
 System.*out*.println("Withdrew: "+ amount + ", New Balance: "+balance);  
 }else {  
 System.*out*.println("Insufficient Funds!!!!, Available Balance is "+ balance);  
 }  
 }else {  
 System.*out*.println("Withdrawal amount should be positive!!!");  
 }  
 }  
  
 //=============================================================================  
  
 //Method to display account details  
 public void displayAccountDetails(){  
 System.*out*.println("Account Number : "+ accountNumber);  
 System.*out*.println("Current Balance : "+ balance);  
 }  
}

package bankAccountQuestion;  
  
public class Main {  
 public static void main(String[] args) {  
 //Creating an object of bankAccount Class  
 BankAccount account = new BankAccount(123,1200);  
  
 //Displaying the initial account details  
 account.displayAccountDetails();  
  
 //Deposit operation  
 account.deposit(300);  
  
 //withdrawing operation  
 account.withdraw(500);  
  
 //trying to withdraw more amount than balance  
 account.withdraw(1800);  
  
 //trying to deposit the negative amount  
 account.deposit(-300);  
  
 //Displaying the final acc details  
 account.displayAccountDetails();  
 }  
}

# create a java program which initializes an array with 3 elements and attempts to print the element at 3rd index.

Write the code and explain what happens when the program is executed.

Why does it results in an exception?

array out of boundry exception will occur because array index start with 0. 3 element means 0,1,2 we are accessing 3 index which is not possible

Which type of exception is thrown?

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 at arrayIndexOutOfBoundExample.ArrayIndexOutOfBoundExample.main(ArrayIndexOutOfBoundExample.java:8)

package arrayIndexOutOfBoundExample;  
  
public class ArrayIndexOutOfBoundExample {  
 public static void main(String[] args) {  
 int[] numbers = {10,20,30};  
  
 //accessing the 3rd index element  
 System.*out*.println("Element at 3rd index: "+ numbers[3]);  
 System.*out*.println("Code to be executed after");  
 }  
}

Explanation:

1. The array numbers have three elements with indices 0, 1 and 2
2. Attempting to access number[3] causes the program to throw the exception which is ArrayIndexOutOfBound because index 3 is out of bounds.

How can it be handled?

package arrayIndexOutOfBoundExample;  
  
public class ArrayIndexOutOfBoundExample {  
 public static void main(String[] args) {  
 int[] numbers = {10,20,30};  
  
 try {  
 //accessing the 3rd index element  
 System.*out*.println("Element at 3rd index: "+ numbers[3]);  
 }catch (ArrayIndexOutOfBoundsException e){  
 //Handling the exception  
 System.*out*.println(e.getMessage());  
 }  
 System.*out*.println("Code to be executed after");  
 }  
}

o/p:

Index 3 out of bounds for length 3

Code to be executed after

Explanation:

TRY block: the code which might throw exception is placed inside the try block.

CATCH block: The ArrayIndexOutOfBoundException is caught and handled inside the catch block with proper message to user.

After handling the exception, the program continues to execute

NULLPOINTEREXCEPTION

package npE;  
  
public class NullPointerExample {  
 public static void main(String[] args) {  
 String str = null;  
  
 //Attempting to call a method on a null object  
 System.*out*.println("Length of the String: "+ str.length());  
 }  
}

o/p:

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "str" is null

at npE.NullPointerExample.main(NullPointerExample.java:8)

EXPLANATION:  
\* The String variable str is initialized to null

* When the program tries to call the length() method on str, it is throwing nullpointerException because null does not have a method to execute.

SOLUTION for normal flow of execution

TASK\*\*\*\*\*\*

#Throw Keyword:

Purpose:

It is used to explicitly throw an exception in a method or block of code

SYNTAX:  
throw new ExceptionType(“Error Message”)

package throwExample;  
  
public class ThrowExample {  
 public static void checkAge(int age){  
 if (age < 18){  
 throw new IllegalArgumentException("Age must be 18 or higher to vote!!");  
 }  
 }  
  
 public static void main(String[] args) {  
 *checkAge*(13);  
 }  
}

throws Keyword:

Purpose:

Declares exceptions that a method can throw to its caller.

package throwsExample;  
  
public class ThrowsExample {  
 public static void riskMethod() throws ArithmeticException{  
 int result = 10/0;  
 System.*out*.println("Result: "+ result);  
 }  
  
  
 public static void main(String[] args) {  
 try {  
 *riskMethod*(); // Caller handles the exception  
 } catch (ArithmeticException e) {  
 System.*out*.println("Exception Occured: "+ e.getMessage());  
 }  
  
 }  
}

|  |  |  |
| --- | --- | --- |
|  | Throw | Throws |
| Purpose | Used to explicitly throw an exception | Declares the exception a method might throw |
| Usage | Used in method body | In the method signature |
| Number of Exception | Only one exception can be handled at a time | Can declare multiple exception(comma seperation) |

TRY-CATCH:

Syntax:

try{

// Code that may throw an exception

}catch(ExceptionType e){

//Handle the exception

}

Custom Exceptions:

Steps for creating custom Exception:

1. Create a class(InsufficientFundsException) and extend it with Exception
2. Add a constructor
3. Throw and handle the exception