Create a BankAccount class that has private fields accountNumber and balance.
Add a 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 bankAccountProgram;

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 the account
    public void deposit(double amount){
        if(amount > 0)  {   // Check if the deposit amount is
positive
            balance += amount; //Add the amount to the balance
            System.out.println("Deposited: "+amount+ ", New
Balance: "+balance);
        }else {
            System.out.println("Deposit amount should be
positive");
        }
    }

    //Method to withdraw money into the account
    public void withdraw(double amount){
        if(amount > 0) {  // Check if the withdrawal amount is
positive
            if(balance >= amount){ //Check if the suffiecient
funds are available
                balance -= amount; //Deducting the amount from
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 the account details
    public void displayAccountDetails(){
        System.out.println("Account Number: "+ accountNumber);
        System.out.println("Current Balance: "+ balance);
    }
}
```

```
package bankAccountProgram;

public class Main {
    public static void main(String[] args) {
        // Create an object of BankAccount class
        BankAccount account= new BankAccount(1234,1000);

        // Display initial account details
        account.displayAccountDetails();

        //Performing a deposit operation
        account.deposit(500);

        // Performing the withdraw operation
        account.withdraw(300);

        //Attempt to withdraw more than balance
        account.withdraw(1500);

        // Attempting to deposit the negative amount
        account.deposit(-60);

        // Displaying the final account details
        account.displayAccountDetails();
    }
}
```

Exceptions:

1. What are Java Exceptions?

   An exception is an event that disrupts the normal flow of program execution.

   It represents runtime issues, like

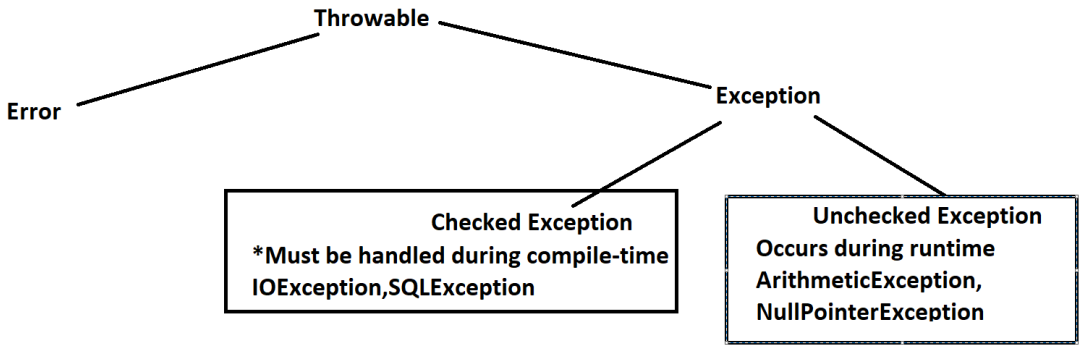
   - Accessing an array index out of bounds
   - Division by zero
   - File Handling errors*

2. Difference Between Errors and Exceptions

	Error	Exception
Definition	Serious issues beyond application control	Issues caused by mistakes in the application code
Control	Unrecoverable by the application	Recoverable using application code
Example	OutOfMemoryError, StackOverFlowError	NullPointerException, IOException
Handling	Not possible	Handled by try-catch blocks

Exception Hierarchy (Throwable (Java SE 21 & JDK 21))

Java Exceptions are part of the Throwable class:



Types of Exceptions:

1. Built-in Exceptions:
   - Checked Exception and Unchecked Exception
2. User-Defined Exceptions:
   We can create custom exceptions by extending the Exception class.
   These are useful when built-in exceptions don’t meet specific requirements

   - Methods to Print Exception Information

1. printStackTrace() : Prints detailed information about the exception and its origin
2. toString(): Provides short description of the exception, showing the class name and message.
3. getMessage(): Displays only the message associated with the exception