/** A BankAccount object has private fields to keep track of its current balance, the number of transactions performed and the Bank in which it is an account, and public methods to access those fields appropriately. */

public abstract class BankAccount {
    private int balance = 0;          // Account balance (whole dollars)
    private int transactionCount = 0; // Number of transactions performed.
    private Bank issuingBank;          // Bank issuing this account

    protected BankAccount( int initialBalance, Bank issuingBank )
    throws InsufficientFundsException {
        this.issuingBank = issuingBank;
        deposit( initialBalance );
    }

    protected int getTransactionFee() {
        return 0;
    }

    protected Bank getIssuingBank() {
        return issuingBank;
    }

    public int withdraw( int amount )
    throws InsufficientFundsException {
        incrementBalance( -amount - getTransactionFee() );
        countTransaction();
        return amount;
    }

    public int deposit( int amount )
    throws InsufficientFundsException {
        incrementBalance( amount - getTransactionFee() );
        countTransaction();
        return amount;
    }

    public int requestBalance() 
    throws InsufficientFundsException {
        incrementBalance( -getTransactionFee() );
        return balance;
    }

    protected void incrementBalance( int amount )
    throws InsufficientFundsException {
    }

    protected abstract Bank getIssuingBank();

    protected abstract class BankAccountException {
    }

    protected BankAccountException( ) {
    }

    protected BankAccountException( String msg ) {
    }

    protected BankAccountException( Throwable cause ) {
    }

    protected BankAccountException( String msg, Throwable cause ) {
    }
Listing 7.2: BankAccount.java

```java
113    countTransaction();
114    return getBalance();
115
116    /**
117     * Get the current balance.
118     * Does NOT count as a transaction.
119     *
120     * @return current account balance
121     */
122    
123    public int getBalance()
124    {
125        return balance;
126    }
127
128    /**
129     * Increment account balance by given amount.
130     * Also increment issuing Bank's balance.
131     * Does NOT count as a transaction.
132     *
133     * @param amount the amount of the increment.
134     *
135     * @exception InsufficientFundsException when appropriate.
136     */
137    
138    public final void incrementBalance(int amount)
139    throws InsufficientFundsException
140    {
141        int newBalance = balance + amount;
142        if (newBalance < 0) {
143            throw new InsufficientFundsException( "for this transaction" );
144        }
145        balance = newBalance;
146        getIssuingBank().incrementBalance(amount);
147    }
148
149    /**
150     * Get the number of transactions performed by this account. Does NOT count as a transaction.
151     *
152     * @return number of transactions performed.
153     */
154    
155    public int getTransactionCount()
156    {
157        return transactionCount;
158    }
159
160    /**
161     * Increment by 1 the count of transactions, for this account and for the issuing Bank.
162     * Does NOT count as a transaction.
163     *
164     * @exception InsufficientFundsException when appropriate.
165     */
166    
167    public void countTransaction()
168    throws InsufficientFundsException
169    {
170        transactionCount++;
171        this.getIssuingBank().countTransaction();
172    }
173
174    /**
175     * Action to take when a new month starts.
176     *
177     * @exception InsufficientFundsException thrown when funds on hand are not enough to cover the fees.
178     */
179    
180    public abstract void newMonth();
181
182    /**
183     * Get the current balance after the transaction.
184     * Does NOT count as a transaction.
185     *
186     * @return current account balance after the transaction.
187     * @exception InsufficientFundsException thrown when funds on hand are not enough to cover the fees.
188     */
189    
190    public int getTransactionBalance()
191    {
192        return getBalance() + transactionBalance;
193    }
194
195    /**
196     * Get the account balance after the transaction.
197     * Does NOT count as a transaction.
198     *
199     * @return account balance after the transaction.
200     * @exception InsufficientFundsException thrown when funds on hand are not enough to cover the fees.
201     */
202    
203    public int getIssuingBankAccountBalance() {
204        return getBalance() + getIssuingBank().getBalance();
205    }
206```