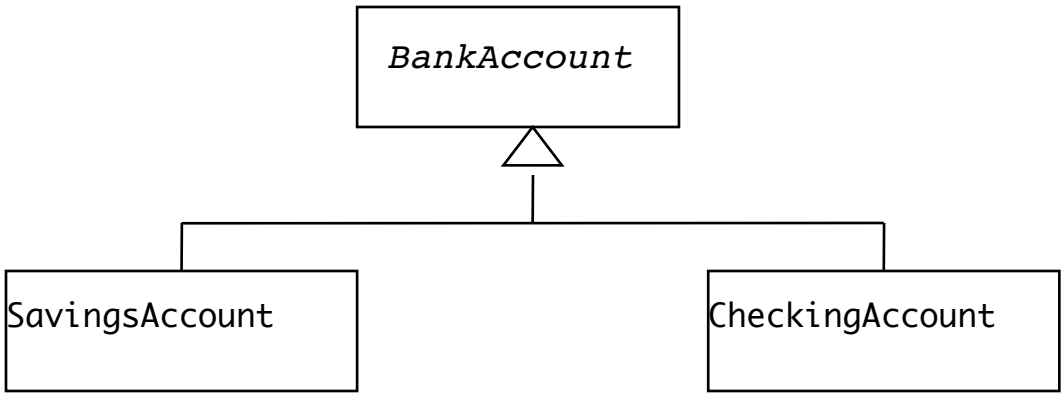
An Example of a Simple Class Hierarchy




```
/**
 * Representation for a bank account
 *
 * @author Russell C. Bjork
 * @version November 18, 2009
 */
public abstract class BankAccount
{
    private final int accountNumber;
    protected Customer owner;
    protected int currentBalance;     // Represented in cents
    private static int nextAccountNumber = 1;

    /**
     * Constructor for objects of class BankAccount
     *
     * @param owner the owner of this account
     *
     * The account number will be set to the first available unused number
     * The balance will be set to zero
     */
    public BankAccount(Customer owner)
    {
        accountNumber = nextAccountNumber ++;
        this.owner = owner;
        owner.addAccount(this);
        currentBalance = 0;
    }

    /**
     * Deposit money
     *
     * @param amount the amount to deposit (in cents)
     */
    public void deposit(int amount)
    {
        currentBalance += amount;
    }
```

```
    /**
     * Withdraw money
     *
     * @param amount the amount to withdraw (in cents)
     * @exception IllegalArgumentException if insufficient balance on hand
     */
    public void withdraw(int amount)
    {
        if (currentBalance < amount)
            throw new IllegalArgumentException("Insufficient balance for withdrawal");
        currentBalance -= amount;
    }

    /**
     * Report current balance.
     * @return current balance, formatted neatly as dollars and
     *         cents, with a dollar sign and decimal point
     */
    public String reportBalance()
    {
        String result = "$" + currentBalance/100 + ".";
        int cents = currentBalance % 100;
        if (cents < 10)
            result += "0";
        result += cents;
        return result;
    }

    /** Accessor for account number
     *
     * @return account number for this account
     */
    public int getAccountNumber()
    {
        return accountNumber;
    }
}
```

```
**
 * Representation for a checking account
 *
 * @author Russell C. Bjork
 * @version November 18, 2009
 */
public class CheckingAccount extends BankAccount
{
    /**
     * Constructor for objects of class CheckingAccount
     *
     * @param owner the owner of this account
     *
     * The account number will be set to the first available unused number
     * The balance will be set to zero
     */
    public CheckingAccount(Customer owner)
    {
        super(owner);
    }

    /**
     * Withdraw money - override of method inherited from BankAccount.  If the
     * balance is insufficient, but the customer has a savings account with
     * a sufficient balance, withdraw the money from savings instead; otherwise
     * use the method inherited from the superclass.
     *
     * @param amount the amount to withdraw (in cents)
     * @exception IllegalArgumentException if insufficient balance on hand
     */
    public void withdraw(int amount)
    {
        if (currentBalance < amount && owner.getSavingsAccount() != null &&
                owner.getSavingsAccount().currentBalance >= amount)
            owner.getSavingsAccount().withdraw(amount);
        else
            super.withdraw(amount);
    }
}
```

```
/**
 * Representation for an interest-bearing savings account
 *
 * @author Russell C. Bjork
 * @version November 18, 2009
 */
public class SavingsAccount extends BankAccount
{
    private static double annualInterestRate;

    /**
     * Constructor for objects of class SavingsAccount
     *
     * @param owner the owner of this account
     *
     * The account number will be set to the first available unused number
     * The balance will be set to zero
     */
    public SavingsAccount(Customer owner)
    {
        super(owner);
    }

    /**
     * Pay interest for one month.
     */
    public void payInterest()
    {
        if (currentBalance >= MINIMUM_AMOUNT_FOR_INTEREST)
            currentBalance += (int) (currentBalance * annualInterestRate / 12.0);
    }

    /**
     * Modify the interest rate
     *
     * @param newRate the new annual interest rate
     */
    public static void setAnnualInterestRate(double newRate)
    {
        annualInterestRate = newRate;
    }

    /** The minimum balance an account can have and still receive interest
     */
    public static final int MINIMUM_AMOUNT_FOR_INTEREST = 500;
}
```