

APPS 105S – Computer Fundamentals

Assignment #3: Methods

Winter 1999

(To be completed for your lab period January 25/26.)

Objective

To introduce methods as a way to organize your program, and to practice the basics of Java programming.

1 A Bank Account Management Program

You are to construct a program that is to be used to keep track of the balances of two bank accounts: a savings account and a checking account. The program will allow for deposits, withdrawals and interest payments to be made on either account.

To begin with, the program will initialize the balance of each account to \$0. The program will then continuously prompt the user to select an action. The choices will be to make a deposit, to make a withdrawal, to calculate and deposit interest, or to quit the program. If either of the first three options are selected, the user is asked which account the action is applied to. **Each of the deposit, withdrawal and interest calculations actions should be implemented as a separate method that takes the current balance of the account as one of its arguments and returns the modified balance as its result.**

The *deposit method* will prompt for a deposit amount, verify that it is a positive value, and then return the new account balance. The *withdrawal method* will prompt for a withdrawal value, but will not permit the withdrawal if the amount is negative or if it is greater than the balance of the account.

The *interest method* will calculate the interest earned on the account and return the new balance. Since interest should only be calculated at regular intervals, the interest calculation would be used only by bank employees. To prevent an unauthorized user from asking for interest, *this command should be protected by a numerical password*. The password number can be a constant hard-coded into your program; when the user requests an interest calculation, your program should prompt the user for the password, and check it.

Separate interest rates for the two accounts can be hard-coded in the program as constants. Pass the appropriate value to the interest calculation method as an argument. To calculate the interest,

apply the interest rate to the *lowest account balance since the last time interest was applied*. The bank employee may ask for interest to be credited as often as desired.
An example of the use of the program is shown below. The program should accept either upper or lower case letters to specify termination, e.g., either Q or q for (Q)uit.

Bank Account Management Program

```
Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? D
(S)avings or (C)hecking? s
Amount of deposit? 1200
Savings account balance: $1200.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
Please enter special password: 76567
Interest Access Allowed.
(S)avings or (C)hecking? c
Chequing account balance: $0.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? d
(S)avings or (C)hecking? c
Amount of deposit? 2000
Chequing account balance: $2000.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
Please enter special password: 77887
Interest Access Denied.

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
Please enter special password: 76567
Interest Access Allowed.
(S)avings or (C)hecking? s
1.5% interest applied to $0.00 = $0.00
Savings account balance: $1200.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
Please enter special password: 76567
Interest Access Allowed.
(S)avings or (C)hecking? s
1.5% interest applied to $1200.00 = $18.00
Savings account balance: $1218.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
Please enter special password: 76567
Interest Access Allowed.
(S)avings or (C)hecking? s
1.5% interest applied to $1218.00 = $18.27
```

```
Savings account balance: $1218.27

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? q
Savings account balance: $1187.27
Chequing account balance: $2000.00
End of Program
```

2 Hints and Notes

The bank account should not hold fractions of a penny after applying interest, and users should not be allowed to deposit or withdraw fractions of a penny. To properly account for this, store the bank account balance as an integer representing the number of cents in the account (rather than the number of dollars). However, when printing the account balance, it must be printed in dollars (don't worry if you can't print out all of the trailing zeros after the decimal). Likewise, deposits and withdrawals should be done in dollar units. Inside your program, convert to cents by multiplying by 100 and using the `Math.floor()` function to remove any fractional part, and convert back to dollars by dividing by 100.

In class, I presented the routines `StdIn.getInt()` and `StdIn.getDouble()`, which return an `int` and `double` type, respectively. In this program, you must also use `StdIn.getChar()`, which returns a `char` type. Just like `int` and `double` variables, you can create `char` variables. These `char` variables can be compared to other `char` variables, or compared to single characters which you must place inside single quotes in your program. The `Average.java` program from lab 2 used `char` variables, and another example follows:

```
// This program writes a prompt to the screen,
// asking the user to type Q to quit. When the
// user finally types Q and ENTER, the program
// prints another message and then quits.
class PromptForQChar
{
    public static void main( String[] args )
    {
        char inputChar;

        do
        {
            System.out.print("Enter Q to quit: ");
            inputChar = StdIn.getChar();
        } while( inputChar != 'Q' );

        System.out.println("We're done now, thanks for typing Q.");
    }
}
```

3 Optional Part

Add extra menu options to close and open the accounts (of course, the bank password is required for these transactions). Do not allow any operation on a closed account except open, and do not allow an already opened account to be reopened. Add an aging feature to the accounts: if one account is neglected in preference to the other one for more than, say, 10 transactions, have the bank close it automatically.

4 Java Keywords

The following *keywords* are used by the Java language and cannot be used for variable names.

abstract	default	if	private	throw
boolean	do	implements	protected	throws
break	double	import	public	transient
byte	else	instanceof	return	try
case	extends	int	short	void
catch	final	interface	static	volatile
char	finally	long	super	while
class	float	native	switch	
const	for	new	synchronized	
continue	goto	package	this	

There are other names that are used by the software packages that come with Java, called the *Java Packages*. You should avoid using these names as well, to avoid confusion with the built-in names. A complete list would be too long to include here, but you can find them in the online Java Platform Core API Specification manual:

<http://www.eecg.toronto.edu/~clarke/jdk1.1.5/docs/api/>