Portfolio element – Smalltalk & Pharo

Unit	Programming languages: principles and design (6G6Z1110) Programming languages – SE frameworks (6G6Z1115)
Lecturer	Rob Frampton
Week	3
Portfolio element	Smalltalk/Pharo (15% of coursework)

Assignment

For this assignment, you will implement a simple bank account system in Pharo.

a) In a package named `BankAccount`, create the following classes:

`Account` class

- Create instance variables `balance` and `interestRate`, and getters for both
- Create method `credit:` with a single argument called `amount`, which adds `amount` to the balance.

`SavingsAccount` class

- Inherits from the `Account` class
- Initialises `balance` to 0 and `interestRate` to 0.015 upon creation

`CurrentAccount` class

- Inherits from the `Account` class
- Initialises `balance` to 0 and `interestRate` to 0.005 upon creation
- Has method `debit:` with a single argument called `amount`, which subtracts `amount` from the balance **if the balance will not become negative** – otherwise, it does nothing.

The following code should test your work:

```
| c |
c := CurrentAccount new.   "Create a new current account"
c credit: 100.             "Pay in £100"
c debit: 75.               "Withdraw £75"
c debit: 75.               "Try to withdraw, but fails due to low balance"
c balance                  "Returns 25"
```

The result should be **25**.

b) Implement a method on the `Account` class named `predictBalanceAfterYears:` with a single argument called `years`. This should return a prediction of the balance after a number of years using the formula:

$$forecast = balance * (1 + interestRate)^{years}$$

Hint: you can raise a number to a power by sending it the `raisedTo:` message

The following code should test your work:

```
| s |
s := SavingsAccount new.          "Create a new savings account"
s credit: 600.                    "Pay in £600"
s predictBalanceAfterYears: 10    "Returns 696.3244950150892"
```

The result should be approximately **696.32**.

c) Implement a method on the `Account` class named `yearsUntilAmount:` with a single argument called `amount`. This method should compute the predicted balance in a loop, increasing the year each time (starting from zero), until the balance reaches `amount`. It should then return the number of years for that balance to be reached.

Note: there are analytical solutions to this problem which do not require a loop, but for the purposes of this exercise, please implement it with a loop.

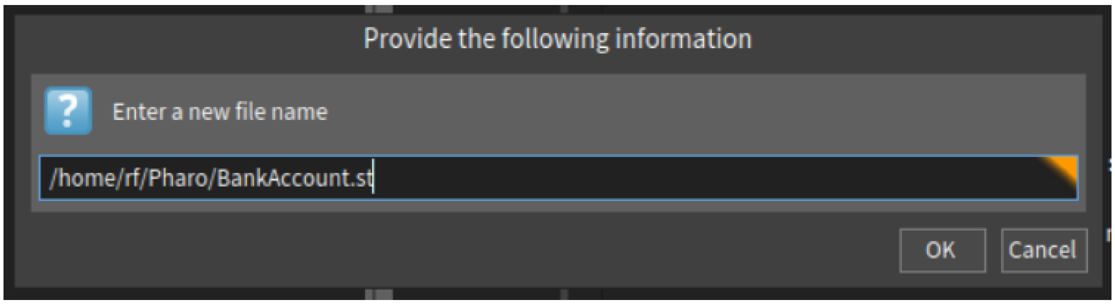
The following code should test your work:

```
| c |
c := CurrentAccount new.          "Create a new current account"
c credit: 300.                    "Pay in £300"
c yearsUntilAmount: 400           "Find number of years until we have £400"
```

The result should be **58**.

Submission

When you are ready to submit, right-click on the `BankAccount` package, click “File Out”, and click “Choose another name”. Enter a path in your home directory. The file you submit should be named **BankAccount.st**.


Provide the following information

Enter a new file name

/home/rf/Pharo/BankAccount.st