**9-2-16**

**Intro to OOP/UML**

Class AccountHolder

{

// Class level variables

static double calcInterestRate;

double balance;

// Constructor

Public AccountHolder (double balance)

{

// Check for positive balance

this.balance = balance >= 0 ? balance: 0;

}

// Setter methods

public void deposit (double newbalance)

{

balance += new balance;

}

public void withdrawal (double newbalance)

{

/\* Impose limit – don’t go below 100 with current balance, also check for balance below $500- issue penalty if so. If balance is not threatened

Balance = newbalance;\*/

}

// Getter method

@override

public String toString()

{

return String.format(“$%.2f“, balance);

}

}

Class AccountHolderTest

{

double balance;

1. // Prompt user for initial balance
2. AccountHolder bankobject = new AccountHolder (balance);
3. // Prompt user for deposit
4. bankobject.deposit (balance);
5. // Prompt for withdrawal
6. // Hard code amount for annualInterest
7. AccountHolder.calcInterestRate = .04;
8. // Print out 12-month report
9. system.out.println (“Month balance = “ + bankobject);

}

**Notes:**

Constructor:

Default

Default-no argument

Parameterized

If you have inheritance, ALWAYS use a CONSTRUCTOR

CEO:

Condition ? true part : false part

if balance >= 0

this.balance = balance

else

this.balance = 0

bankobject1: ID, name, acct #, calcIntRate - .04

bankobject2: ID, name, acct #, annualInterestRate - .04

**UML:**

***Class Name***

*Data members*

*Method members*

+ public

- private

# protected

Dependency (Relationships)

Or Association (Relationship)

**AccountHolder:**

- annual IntRate : double (static)

- balance : double

+ AccountHolder (double)

+ monthly IntRate ( ) : void

+ withdrawals (double) : double