

CS 125 Spring 2009 Exam 1 Review

Determine the output for the following operations:

7 / 4	=	1
1 / 2.0	=	0.5
1.0 / 2	=	0.5
-7 / 2	=	-3

Evaluate the following statements & determine what the output would be:

int x = 10;	x	y	z
int y = 0;	10	?	?
int z = 4;	10	0	?
x++;	10	0	4
z = z + x;	11	0	4
x++;	11	0	15
y--;	12	0	15
z = z + y;	12	-1	15
System.out.println(x);	12	-1	14
System.out.println(y);	prints:		
System.out.println(z);	12		
	-1		
	14		

Create a class called **BankAccount**. It will need an instance field for its balance & another for an interest rate. It should have two constructors, one that sets the initial balance & interest rate to zero & another that sets them based on parameters. It should have the following methods:

- getBalance** - returns the current balance
- deposit** - add a specified amount to the balance
- withdraw** - removes a specified amount from the balance
- setInterest** - used to change the interest rate to a new amount
- getInterest** - returns the current interest rate
- applyInterest** - calculates the interest earned & adds it to the balance

Follow proper naming, coding and documenting conventions. Try to do write the class out on paper first without referring to other code files or the compiler for help.

```
// note: in some places there is more than one possible way to write Java code
// that satisfies the problem statement
```

```
/**
 * models a savings account
 */
public class BankAccount
{
    private double balance;
    private double interest;

    /**
     * default constructor - sets balance & interest to zero
     */
    public BankAccount()
    {
        balance = 0;
        interest = 0;
    }

    /**
     * constructor - sets balance & interest to given amounts
     * @param b - the initial balance
     * @param i - the initial interest rate
     */
    public BankAccount(double b, double i)
    {
        balance = b;
        interest = i;
    }

    /**
     * get the current funds
     * @return the current balance
     */
    public double getBalance()
    {
        return balance;
    }

    /**
     * get the current interest rate
     * @return the current interest rate
     */
    public double getInterest()
    {
        return interest;
    }

    /**
     * add the given amount to the balance
     * @param x the amount to add
     */
    public void deposit(double x)
    {
        balance = balance + x;
    }
}

// continued on next page
```

could also have used this instead:

```
balance += x;
```

```

/**
 * deduct the given amount from the balance
 * @param x the amount to remove
 */
public void withdraw(double x)
{
    balance = balance - x;
}

/**
 * change the interest rate to the given amount
 * @param i the new interest rate to use
 */
public void setInterest(double i)
{
    interest = i;
}

/**
 * determine the interest earned & add it to the balance
 */
public void applyInterest()
{
    balance = balance + (balance * interest);
}
}

```

could also have used this instead:

```
balance -= x;
```

could also have used this instead:

```
balance += (balance * interest);
```

Create a class called **BankAccountTester**. It should have a main method that creates a new **BankAccount** (from above), set the interest rate to 10%, deposit 100 into the account, apply the interest, withdraw 20 & then print out the final balance.

Follow proper naming, coding and documenting conventions. Try to do write the class out on paper first without referring to other code files or the compiler for help.

```

/**
 * tests out out BankAccount code
 */
public class BankAccountTester
{
    /**
     * main method - program starts here
     * @param args - still haven't learned about these yet...
     */
    public static void main(String[] args)
    {
        BankAccount myAccount = new BankAccount();
        myAccount.setInterest(0.1);
        myAccount.deposit(100);
        myAccount.applyInterest();
        myAccount.withdraw(20);

        System.out.println( myAccount.getBalance() );
    }
}

```

could also have replaced the last statement with:

```
double funds = myAccount.getbalance();
System.out.println(funds);
```

What is the significance of the **main** method? Why do we need one?

The main method is the start of a program. If we didn't have one, the computer would have a lot of code, (instructions), but no indication as to what order to execute them in.

What is the primary difference between an **int** and a **double**?

An int represents an interger whereas a double represents a decimal number.

What are instance fields & why are they usually **private**?

Instance fields hold the data that objects need to keep track of. They are generally private because: 1) allowing just anyone to change them at any time might have undesirable effects & 2) it reduces the amount of information other programmers need to use objects of the given class (encapsulation).

Define (with regards to programming) **abstraction**?

Abstraction involves reducing a problem or a model to its essential features. We use abstraction as part of Object Oriented Design to design classes.

What is the relationship between a **class** & an **object** in Java?

Objects are entities we use in our programs – they consist of data (instance fields) that hold information & methods (sequences of instructions) that are used to manipulate that data.

Individual objects are instances (specific examples) of various classes. A class is a programmer defined data type that defines the data & methods used by objects of that class.

To build a new class we code something like: `public class BankAccount ...`

To build a new object we might code: `BankAccoount savings = new BankAccount();`