

# 1.00/1.001

## Introduction to Computers and Engineering Problem Solving

### Quiz 1 / October 7, 2005

<b>Name:</b>	
<b>Email Address:</b>	
<b>TA:</b>	
<b>Section:</b>	

You have 90 minutes to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary files have already been imported.

Good luck.

<i>Question</i>	<i>Points</i>
<b>Question 1</b>	<b>/ 15</b>
<b>Question 2</b>	<b>/ 25</b>
<b>Question 3</b>	<b>/ 10</b>
<b>Question 4</b>	<b>/ 40</b>
<b>Question 5</b>	<b>/ 10</b>
<b>Total</b>	<b>/ 100</b>

<b>HW1</b>	<b>HW2</b>	<b>HW3</b>

### Question 1. Data variables / operators (15 points)

What is the value of 'i' when the following statements are executed:

1. `int i = -4 + 8/3 + 1%2;` i =

2. `double i = -4 + 8/3 + 1%2;` i =

3. `int i = -4 + 8.0/3 + 1%2;` i =

4. `float i = -4 + 8.0F/3 + 1%2;` i =

5. `int i = -4 + 8/(3 - 1)%2;` i =

## Question 2. True / False and short answer (25 points)

Answer the following questions about Java by circling TRUE or FALSE, or by circling the correct answer(s) from the multiple choices as appropriate.

1. A static data member is shared by all instances of that class.  
**TRUE** FALSE
2. A static data member's value can never change.  
TRUE **FALSE**
3. A static data member can be declared as public or private.  
**TRUE** FALSE
4. A private non-static data member of a class may be accessed from any non-static method of that class.  
**TRUE** FALSE
5. A local variable declared in one method may be accessed by all the methods beneath it in the same class.  
TRUE **FALSE**
6. Which of the following statements is true for the local variable `temp` as shown in the code fragment below?

```
int i = 100;
while(i > 0){
    int temp = 0;
    i--;
//rest of implementation hidden
}
```

[Note: zero or more options may be correct]

- a. `temp` may be accessed anywhere in that method.
- b. **`temp` may only be accessed within the loop.**
- c. **`temp` is “destroyed” when the loop terminates**

7/8. `Bus` and `Engine` are classes in the same package. Code fragments from each class are shown below.

```

//Code from Bus.java
public class Bus{
    //data members
    Engine myEngine;
    private String myNumber;

    //rest of implementation hidden . . .
}

//Code from Engine.java
public class Engine{
    //data members
    Bus myBus;
    private String myNumber;

    //rest of implementation hidden . . .
}

```

7. The statement `engine1.myNumber = "E1251F045";` will compile if
- engine1 is a reference to an Engine object within the Engine class.**
  - engine1 is a reference to an Engine object either within the Engine class, or within the Bus class.
  - engine1 is a reference to an Engine object in any class that is in the same package as Engine.
  - engine1 is a reference to an Engine object in any class.
8. Within a method of the Bus class is the statement:  
`myEngine.myBus=this;`  
This statement will cause an error during compiling.

TRUE

**FALSE**

### Question 3. Recursion (10 points)

Given the method below:

```
public static int method(int k)
{
    if (k == 1)
        return 0;
    return (method(k-1) + 2);
}
```

Answer the following questions:

1. If the method is invoked with argument  $k = 3$ , what will it return?

**Four**

2. If the method is invoked with argument  $k = 1$ ?

**Zero**

Extra credit (5 points). Can you state in words what this method calculates?

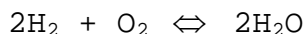
**Returns the (k-1)th even number  
(or the kth even number starting from zero)**

#### Question 4. ArrayList (40 points)

You are trying to model chemical equations using Java classes. So far you've come up with a simple Component class that describes all the species in a chemical reaction.

```
public class Component {
    public String name;
    public String symbol;
    public int coefficient;
    public double concentration; // Assume that all
//components have the same units for concentrations
}
```

Consider the reaction



An example of a component would be "H<sub>2</sub>O", otherwise known as water. The values of the data members of the Component class would be as follows.

```
name: "Water",
symbol: "H2O",
coefficient:2, //The number that goes before the
//component's symbol in the equation
concentration:5.5 //Concentration is randomly assigned here
//; it isn't obvious from the reaction.
```

You now need to write a Reaction class that can describe a general reaction equation. Put your code in the blank areas on the following pages.

```
public class Reaction {
```

```
/* 1. Data Members (all private members)
 * Declare an ArrayList of Components for reactants.
 * Declare another ArrayList of Components for products.
 * For example, in the above reaction, H2 and O2 are the
 * reactants and H2O is the product.
 */
```

```
    ArrayList<Component> reactants;
```

```
    ArrayList<Component> products;
```

```
/* 2. Complete the constructor for an empty Reaction;
 * initialize the two ArrayLists
 */
```

```
public Reaction () {
```

```
    reactants = new ArrayList<Component>();
```

```
    products = new ArrayList<Component>();
```

```
}
```

```

/* 3. Complete the method, addProduct(), which takes in a
 * given Component and adds it to the product ArrayList.
 */

public void addProduct(Component product)
{
    products.add(product);
}

```

```

/* 4. Complete the method calculateK(), which calculates
 * the equilibrium constant (commonly denoted as K) for
 * the given reaction. As an example the equilibrium
 * constant, K, for the reaction:
 *
 *      aA + bB ⇌ cC + dD + eE
 *
 * is defined as  $K = \frac{[C]^c * [D]^d * [E]^e}{[A]^a * [B]^b}$ , where
 *
 * [A] represents the concentration of component A,
 * and 'a' represents the coefficient of component A.
 * [A]a has to be calculated as the (concentration of A)
 * raised to the power of the (coefficient of A)
 *
 * Assume that the reactant and product ArrayLists have
 * already been filled with the correct values. Also note
 * that concentration and coefficient are data members of
 * the Component class.
 */

public double calculateK () {
    //(i) Define 2 variables 'numerator' and 'denominator'.
    //Decide what data types would be most appropriate for them
    //and what values they must be initialized to.

    double numerator = 1.0;
    double denominator = 1.0;
}

```

```

//(ii) Compute 'denominator' as the product of the
//concentrations of the Components in the reactants
//ArrayList raised to the power of their coefficients.
//For example, denominator = [A]a * [B]b

for (int i =0; i<reactants.size();i++) {
    denominator = denominator * Math.pow(
        reactants.get(i).concentration,
        reactants.get(i).coefficient);

    //Alternatively you can also do the following
    double conc = reactants.get(i).concentration;
    int coeff = reactants.get(i).coefficient;
    denominator = denominator * Math.pow(conc, coeff);

}

//(iii) Compute 'numerator' as the product of the
//concentrations of the Components in the products
//ArrayList raised to the power of their coefficients.
//For example, numerator = [C]c * [D]d * [E]e
for (int i =0; i<products.size();i++) {
    numerator= numerator * Math.pow(
        products.get(i).concentration,
        product.get(i).coefficient);
}

//(iv) Return the equilibrium constant K, which is equal to
//numerator divided by the denominator.
    double K = numerator / denominator;
    return K;

    (Alternatively return numerator /denominator)
}

```

## Question 5. Methods (10 points)

Read the descriptions of the methods given below. For each description, write the method signature that would be used to declare such a method. Do not fill in the body of the method – only write the signature.

EXAMPLE: `doNothing()` can be called only from within this class. It has one `int` argument named `arg`, and returns a `String`. Its signature would be written as:

```
private String doNothing(int arg) {  
    //don't write anything here - leave it empty  
}
```

1. `whoops()` can be called from anywhere in the program, and can be run without creating an instance of the class. It takes one `double` argument named `dub` and returns a `String`.

```
public static String whoops(double dub) {  
}
```

2. `almostDone()` can be called from anywhere in the program, but requires an instance of the class to call it. It takes two arguments, one `Calendar` named `yay`, and one `double` named `sweet`. It does not return a value.

```
public void almostDone(Calendar yay, double sweet) {  
}
```