

# 1.00/1.001

## Tutorial 4

October 3<sup>rd</sup>, 4<sup>th</sup> 2005

# Things to be discussed....

- Quiz 1 Logistics
- Concepts/key ideas you don't understand!
- Scope
- Recursion
- Problem Set 4
- ArrayLists
- Static data and Static methods

# Quiz 1 Logistics

- Friday, October 7 class time (11 to 12:30pm)
- Topics Included
  - Lectures 1 to 11
  - PS 1 to PS 3
- Open Book, Open Notes, **NO** Laptops
- Optional Quiz 1 Review on October 5 (7-9pm).

# Concepts you don't get?

- Anything we need to go over again?
- Main Quiz topics:
  - Data types, variables, operators
  - Control structures (if/else, loops, switch)
  - Classes/objects(including static members)
  - Methods (including scope & access)
  - Arrays, ArrayLists
  - Recursion

# Scope of local variables

```
for (int j = 0; j<5; j++)  
{  
    int k = j;  
}  
System.out.println(k);
```

What's wrong with this code?

# Scope Question

```
for(int y = 0; y < 5; y++)  
{  
    double x = 100.0;  
}  
for (int z = 10; z < 20; z++)  
{  
    int x = 10;  
}
```

Will this code compile?

# Scope Question

```
int x = 100;
for(int y = 0; y < 5; y++)
{
    double x = 100.0;
}
```

Does this compile? Why or why not?

# Scope of Class Members

```
public class OverlappingScope
{
    int num ;
    public OverlappingScope(int num)
    {
        // How would you set the value of the
        // class variable num?
    }
}
```

# Scope : Types of variables

```
public class Tutorial4 {
    private int n=0;
    public static final int CON = 5;

    public Tutorial4(int a)
    {n=a;}

    public static int method1(int b){
        int temp= CON ;
        temp = b++;
        return temp;
    }

    public int method2(int c){
        int temp= n+CON;
        temp = c++;
        return temp;
    }

    public int getn()
    { return n;}
}
```

- In the code, list
  - Instance variables
  - Static (non-Instance) variables
  - Method Arguments
  - Local variables
- What is the scope of variable "*temp*" in the methods `method1()` & `method2()`?  
i.e is the "*temp*" variable of `method1()` same as that of `method2()`?  
Why or why not?
- Will this code compile?

# Recursion

- A divide and conquer approach
- Needs a **Base Case** and a way to solve a bigger problem using solutions to smaller problems

```
// The sum of the first n natural numbers
public static int sum (int n){
    if (n == 0)    // Base Case
        return 0;
    else          // General Case
        return (n + sum(n-1));
}
```

# Recursion Question

```
public static int fact (int n){  
    return (n * fact(n-1));  
}
```

What is wrong here?

# Recursion Question

```
public static int sum ( int n){
    int sumN;
    if (n == 0)
        return 0;
    else
    {
        sumN = n + sum(n-1);
        return (sumN);
    }
}
public static void main( String[] args)
{
    int sumN = sum(10);
}
```

Will this code work? What about sumN?

# Recursion Exercise

Write a recursive method for `fibonacci(n)` where:

$$\mathbf{fib(n) = fib(n-1) + fib(n-2)}$$

Create a public static method called `fib` in a class `Fibonacci`

# Problem Set 4

- Builds on Problem Set 3
  - You can use your own PSet3 solution or the one that we provide you online
- Goals
  - Use inheritance
  - ArrayLists

# Using ArrayLists

- Must import `java.util.*`;
- Common constructors (e.g. of constructor overloading)

```
ArrayList<String> list1 = new ArrayList<String>();  
ArrayList<String> list2 = new ArrayList<String>(20);
```

- Adding to a ArrayList

```
list1.add("Felicia");  
list1.add(3, "Daniel");
```

- Getting things out

```
String TA = list1.get(2);
```

- Other methods (look at javadoc):

```
int noTAs = list1.size();  
list1.remove("Karin");
```

.....

# Exercise : Using ArrayLists

- Create an ArrayList containing the Integer objects that correspond to the numbers 1 to 10. Print the values of the ArrayList & their sum at each step.

# Static Data Members & Methods

```
public class Class1 {
    private int n=0;
    public static final int CON = 5;

    public Tutorial4(int a)
        {n=a;}

    public static int method1(int b){
        //method body;
    }

    public int method2(int c){
        //method body
    }
}

public class Class2 {
    public static void main(String[] args){
        //to complete
    }
} //end of class Class2
```

- In the main() method of class Class2, invoke
  - Method method1() ;
  - Method method2();
  - How can you access variables "n" and "CON" ? (you can add methods in Class1 if required)(Assume both Class1 and Class2 are in default package)

# Now, does all this make sense ?

- Static members:
  - are not associated with any particular instance of the class—one copy shared by all instances
  - are accessible to both static and non-static methods
  - Typically *public* and in most case are declared as *final*.
- Static Methods:
  - may only access static members, **not** instance members
  - may be called using *Classname.methodName(...)* or *objectName.methodName(...)*.

# When to Use Static Methods

- When no access to any instance field is required. Usually one of two scenarios:
  - The method takes in all the information it needs as parameters:  
`Math.pow(double base, double exp)`
  - Or, the method needs access to only static variables.