Method Examples

A rehash
Cleans Up Code

Consider following

```java
public static void main(String[] args)
{
    Shuttle s = new Shuttle();
    s.launchSpaceShuttle();
    s.repairSatellite();
    s.takeCoolDigitalPictures();
}
```

Lots of stuff going on in the methods!
- Would be too messy to have all of this in the main().
Methods Don’t Have to Return

Still useful. Consider this.

```java
public class CoolStuff {
    public static void main(String[] args) {
        CoolStuff cs = new CoolStuff();
        cs.runMyCode();
    }

    public void runMyCode() {
        //all your code here
    }
}
```

Nothing is in main(). Just moved all of its code to the method!
Suppose *later* I want to run something from another program called ExtraCoolStuff.

- Just add it to `main()` as

```java
public class CoolStuff
{
    public static void main(String[] args)
    {
        CoolStuff cs = new CoolStuff();
        cs.runMyCode();
        ExtraCoolStuff ecs = new ExtraCoolStuff();
        ecs.runItDude();
    }
    ...
}
```

Easy to add new code!

Call this modular.

Plug-n-play.
Another Example

- Remember math constant “e”?  
  - $e = 2.71828…$

- Calculate $e$ using 100 terms from  
  - $e = 1/0!+1/1!+1/2!+1/3!+1/4!+…$
public class MathStuff
{
    public static void main(String[] args)
    {
        double e = 0.0;
        for(int i=0; i<100; i++)
        {
            int factorial = 1;
            for(int n=1; n<=i; n++)
            {
                factorial *= n;
            }
            e += 1.0/((double) factorial);
        }
        System.out.println("e = "+e);
    }
}
public class MathStuff
{
    public static void main(String[] args)
    {
        double e = 0.0;
        for(int i=0; i<100; i++)
        {
            MathStuff m = new MathStuff();
            int factorial = m.factorial(i);
            e += 1.0/((double) factorial);
        }
        System.out.println("e = "+ e);
    }
}

public int factorial(int i)
{
    int factorial = 1;
    for(int n=1; n<=i; n++)
    {
        factorial *= n;
    }
    return factorial;
}
Remember sin(x)?

- You wrote code to calculate sin(x).
- Used loops.

Let’s do the same thing for cos(x) but use methods
  - Will use loops just like sin(x).
  - Same calculations
    - Just put loops into methods.
Cos Problem Statement

- Calculate $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$ by adding 100 of these terms.

Solution:
- Need loop that adds 100 terms then quits.
  - i.e., will loop from term=0 to term<100

  ```
  for(int term=0; term<100; term++)
  {
    //put calculations in here…
  }
  ```
What Methods Do We Need?

- For each term need to calculate $x^m$.
  - Do this in a method.
    - Pass it $x$ and $m$. Return a double value.

- For each term need to calculate $m!$
  - Do this in a method
    - Pass it $m$. Return an int value.

- Also need to calculate the sign (+ or –)
  - Do this in a method
    - Pass it the number of the term (e.g., 1, 2, 3…).
    - Return a double value of (1.0 or –1.0) for that term.
public class HomeworkMath
{
    public int factorial(int m)
    {
        int fact = 1;
        for(int n= 1; n<=m; n++)
        {
            fact *= n;
        }
        return fact;
    }
    public double power(double x, int m)
    {
        double xToTheMthPower = 1.0;
        for(int n=1; n<=m; n++)
        {
            xToTheMthPower *= x;
        }
        return xToTheMthPower;
    }
    public double sign(int term)
    {
        return power(-1.0, term);
    }
}
Now Can Call the Methods.

e.g.,

```java
public class TestCosMethods {
    public static void main(String[] args) {
        HomeworkMath hm = new HomeworkMath();
        double powerValue = hm.power(3.0, 2); //returns 3.0^2 (i.e., 9.0)
        int factorialValue = hm.factorial(4); //returns 4! (i.e., 24)
        double signValue = hm.sign(77); //returns (-1.0)^77 (i.e., -1.0)
    }
}
```
public static void main(String[] args) {
    double xValue = 2.0;
    double cos = 0.0;
    HomeworkMath hm = new HomeworkMath();

    for(int term=0; term<100; term++) {
        double powerValue = hm.power(xValue, 2*term);
        double factorialValue = (double) hm.factorial(2*term);
        double signValue = hm.sign(term);

        cos += signValue * powerValue / factorialValue;
    }
}

Much better than putting the methods’ code inside main(). That would be messy!

Didn’t want integer division, so cast factorial to a double.