# Subprograms (methods)

- A method is a  subprogram that can be accessed and used by other programs

- **A method is defined by**

*public static void printSquare (int num, double y){…}*

1) The method's identifier (name)
2) The number of parameters
   - If there are two being passed to the method, two must be received by the method.
   - *e.g. printSquare(3, 4.11111);*
3) The type(s) of parameter(s)
   - the types of parameters received, must match the types sent in the order they are sent
   - *e.g. printSquare(3, 4.11111);*

# 2 types of Methods

Methods can be divided into two groups,

- Commands
- Queries

# 1) Commands

- Commands are called to perform some task
- Once complete, they return to the point at which they were called
- e.g.

```
public static void smaller (int x, int y){
    int sml=y;
    if (x<=y)
        sml = x;
    System.out.println(sml);
}
```

# 2) Queries

- Used to calculate some value
- That value is then returned to the point at which the method was called

```
public static int larger (int x, int y){
    int lrg=y;
    if (x>=y)
        lrg = x;
    return lrg;
}
```

# So….

- What is the main difference between commands and queries?

- Queries **return** a value. Commands do not.

# Invalid Parameters

- A method must communicate to the main method when an invalid parameter is received.

- This is usually done using a Boolean value or an integer that is not used in the set (-1?)

# Commenting Methods

- All methods should contain the following comments:
  - A description of the parameters being received and their purpose
  - What this method will calculate
  - What is returned and why.

# Invoking a method

```
public static void main (String[args[]){
      //invoking a command
      smaller (4, 6);

      //invoking a query
      int number;
      number = larger (4, 8);
      System.out.println( number);
      System.out.println( larger (5,4));
      // since the method larger will return a value, we need to do
    something with that value
}
```

# Method Overloading

- Method Overloading involves creating a multiple number of the same method to accommodate for a variety of parameters that could be potentially passed in.

- The type of / and number of parameters passed is what determines which version of the method is called.

**For Example:**

```
public static void Overload () {
        System.out.println(“This method receives NO parameters”);
}

public static void Overload (int x, int y){
                System.out.println("This method receives integers");
        }

public static void Overload (double x, int y){
                System.out.println("This method receives a DOUBLE and an integer");
        }

public static void main(String[] args) {
                Overload();
                Overload(3,6);    //ok
                Overload(3.23,5);
                Overload(3.2,4.5);   //error here
        }
```

- In the preceding example,

- 3 methods are created using the same method name (i.e. Overload)
  - method 1 receives: no parameters
  - method 2 receives: two integers
  - method 3 receives: a double and an integer
    IN THAT ORDER

- When the method Overload is called
  - Method 1 is called if there are no parameters passed
  - Method 2 is called if there are two integers passed
  - Method 3 is called if there is a double and an integer passed in that order

# Creating Libraries

- You are able store all of your methods in a separate class.
- To invoke methods from a separate class, identify the class that it is located.

# For example

```
public class method {

     public static void smaller (int x, int y){
          int sml=y;
          if (x<=y)
               sml = x;
          System.out.println(sml);
     }

     public static int larger (int x, int y){
          int lrg=y;
          if (x>=y)
               lrg = x;
          return lrg;
     }
}
```

```java
public class Main {

/**
 * @param args
 */
    public static void main(String[] args) {
    // TODO Auto-generated method stub
        int num;
        System.out.println("sad");
        method.smaller(14,3);
        method.smaller(17,38);
        num=method.larger(1,2);
        System.out.println(num);
        System.out.println(method.larger(2, 3));
    }
}
```