CS620
Structured Programming
Introduction to Java

Day 4 - Lecture 1

Introduction to Methods and user input.

Keyboard Input

- Up to now we have declared variables at the start of every program
- Sometimes we don’t know the value of a variable until the program runs, e.g. until a user provides input
- Keyboard input since Java 1.5 is much more straight-forward
- Java 1.5 introduced the Scanner class which simplifies input generation. It can also be used to read from files, keyboard and Strings (among other sources).

Keyboard Input

- To get keyboard input you need three things…

1. The Scanner class is defined in the java.util package. So you must have the following statement at the top of your program

   import java.util.*; OR

   import java.util.Scanner;

2. Need to create an instance of the Scanner class

   Scanner sc = new Scanner(System.in)

3. Need to call one or more Scanner methods to read in data input

Important Methods

- nextInt() // reads in the next token as an int
- nextFloat() // reads in the next token as an float
- nextDouble() // reads in the next token as an double
- nextLine() // reads in the next token as an String
- hasNextInt() //returns true if there is another int to read in
- hasNextFloat() //returns true if there is another float to read in
- hasNextDouble() //returns true if there is another double to read in
- hasNextLine() //returns true if there is another String to read in
- Look at the API for a full list of methods …

Let's take a look at an example

  - GettingInput.java

Using the Scanner class (tool to read input)

```
import java.util.Scanner;  // use this toolkit
```

Scanner is a Class (or type) or a template for creating an object in your program. Think of a template or stencil for making cookies. **Class is to object as template is to cookie.**

The line below uses the **Class constructor** to **create** an object **sc** of type Scanner. What we are doing here is creating an object which will be attached to the standard input (keyboard) and become a software representation for the keyboard in your program (**sc** below)

```
Scanner sc = new Scanner(System.in);
```

Now we can send messages to this newly created object **sc** or use method calls on it to retrieve information from the keyboard. The list of all possible method calls on an object of a certain class is called the class API.

```
int number = sc.nextInt( );
```

Getting user input example.

```java

/**
 * This application demonstrates the use of the Scanner class
 * in order to obtain input from the user of your program.
 *
 * @author Joe Duffin
 * @version 04/09/2013
 */


// you need to import the Scanner class from the java.util library before using it in your code
import java.util.Scanner;

public class GettingInput
{
    public static void main(String []args){

        // Creates a object of type Scanner using the class constructor while
        // passing the System.in vaue to the constructor.
        Scanner myScanner = new Scanner(System.in);

        System.out.println("Please enter your first name and press enter");
        String myName = myScanner.nextLine();

        System.out.println("Hi " + myName);

        System.out.println("Please enter a number and press enter : ");
        String myNumber = myScanner.nextLine();

        System.out.println("you entered the number " + myNumber);

    }
}
```

More details

- Take a look the Scanner class in the Java API
- Let’s take a look at another example

CS141 January 2009 Examination

Write a Java Program that achieves the following:

(i) Asks the user to enter a String representing a typical Irish 12-digit mobile phone number composed of 3 digit international code, 2 digit operator code, and seven digit number. For example 353874123122

(ii) Checks that exactly 12 digits have been entered. The program should prompt the user to enter exactly 12 digits and not progress until they have done so. You may assume that once 12-digits are entered the phone number is valid.

(iii) The program should print the 12 digit number, the international code, the operator code, the seven-digit number and the complete number formatted as shown in the last line of the example below. Example:

Enter a 12-digit Irish mobile phone number:

353874123122

You entered 353874123122

The international code is 353

The operator code is 87

The number is 4123122

The complete telephone number is (353) 87 4123122

```
import java.util.Scanner;

  public class January2009Q3B
  {
    public static void main( String [] args )
    {
      String number, inter, op, remaining; // Create Strings
      Scanner scan = new Scanner( System.in ); // Create a scanner
      System.out.print( "Enter an Irish mobile phone number > " );
      number = scan.nextLine(); // Read in number
      while(number.length()!= 12) // Check if number is not of length 12
      {
              System.out.print( "Enter an Irish mobile phone number > " );
              number = scan.nextLine();
      }

      System.out.println("You entered " + number);
      inter = number.substring(0,3);
      System.out.println("The international code is " + inter);
      op = number.substring(3,5);
      System.out.println("The operator code is " + op);
      remaining = number.substring(5);
      System.out.println("The number is " + remaining );
      System.out.println("The complete number is (" + inter + ") " + op + " " + remaining);

    }
  }
```

Question for you..

- Write a program that asks the user to enter:
  - Name
  - Address
  - Dear of birth
  - Wages
  - No of brothers and sisters

The program should print out the following with the values in { } replaced by the input.

Hi {Name},

You live at {Address}

Your date of birth is {Date of birth}

You earn {wages}

You have {No of brothers and sisters} brothers and sisters

Introducing Methods

- A java method is a bundle of lines of code with a common function.

- When defined in an program it can be used (called) repeatedly.

- Methods also define the behaviour of Classes in object oriented programming (more to follow on this)

Methods encountered to date

- You have encountered a number of different methods in your code so far.

- You have written a main method in your applications and you have used:

System.out.println(“Hello world”);

public static void main(String [] args)

Writing custom made methods

• Writing methods helps to keep program code manageable.

•It allows us to maintain important code or algorithms that are used in repeatedly in different places in our applications.

•It allows the sharing of functionality with other parts of our software system.

Key Features of Methods

- A method must be written or declared before it can be used.

- After it is declared it is used by calling or invoking it in your program.

The line of code below is described as a call to the println method.

```
System.out.println(“Hello World”);
```

Formal parts of methods in Java

Method Declaration:

```
Access_modifiers   return_type  method_name( formal_parameter_list)
{
        statements;
}
```

e.g.

```
public static void printMsg (  )
{

        System.out.println("Hello world");

}
```

It is important to know the form or syntax of a method declaration. The next slides will explain a method declaration by using examples.
The method above is used in your code by typing:

```
printMsg( ); // This line of code will execute the code of the method at this point.
```

Formal parts of methods in Java

Method Declaration:

access_modifiers **return_type** method_name **(formal parameter list)**
{
    statements; // if the return type is not void then you need a **return** statement.
}

Note:: The formal parameter list can be empty or it has comma separated declared parameters as in the example below.

public static **int** addTwoNumbers **(int a, int b)**
{
    return a +b;
}

It is important to know the form or syntax of a method declaration. The next slides will explain a method declaration by using examples.
The method above is used in your code by typing:

Int x = addTwoNumbers(10,27); // This line of code will execute the code of the method at this point.

Method declaration and use of the method.

- The method declaration tells the programmer how the method should be used.

- The declaration is the design of the method and it must be used according to its design.

- For example if a method is designed to take one parameter in the parameter list then writing more than one parameter or none when calling the method will generate and error.

- The number of parameters in the call to the method must match the type and number of parameters in the method declaration.

- There are other rules but it is best to illustrate them with examples.

(1) Taking no parameters and no return type

```
1
2  /**
3   * This program illustrates the essential aspects of declaring a method
4   * and using this method in the main method of the application.
5   *
6   * @author Joe Duffin
7   * @version 4/09/2013
8   */
9  public class MethodSimple1
10 {
11     public static void main(String args[])
12     {
13         int count=0;
14         // This is a method call or a method invocation.
15         // Notice that this method is designed not to take any parameters.
16         myPrintMsg(); // this is a method call or a method invocation.
17
18         while(count <=3){
19
20             myPrintMsg(); // call to execution of the myPrintMsg method
21             count++;
22         }
23     }
24     /**
25      * This is the method declaration
26      */
27     // The first line below is the method signature.
28     // void means that it does not return any values.
29     // Notice also that it does not take any parameters.
30     public static void myPrintMsg()
31     {
32         System.out.println("Hello World");
33     }
34 }
```

You make a call to a method by using its name followed by parentheses. The information between the parentheses is the actual parameter list. In this case the method does not take any parameters and this is consistent with the way it was designed in the formal parameter list of the method signature in the method declaration below.

void means that the method does not return and information to the its calling context.

The formal parameter list of this method is empty so when we use it or call it we do not supply it with an actual parameter.

(2) Takes two parameters and no return type

```
1  /**
2   * This program illustrates the essential aspects of declaring a method
3   * and using this method in the main method of the application.
4   *
5   * @author Joe Duffin
6   * @version 4/09/2013
7   */
8  public class MethodSimple2
9  {
10     public static void main(String args[])
11     {
12         int num1 = 5;
13         int num2 = 6;
14         // This is a method call or a method invocation.
15         // Notice that this method is designed to take two int parameters
16
17         addNumbers(num1, num2); // this is a method call or a method invocation.
18     }
19     /**
20      * This method takes two integers, adds them together and
21      * prints the result to the screen
22      */
23     // The first line below is the method signature.
24     // void means that it does not return any values.
25     // Notice also that the method is designed to take two parameters a and b
26     public static void addNumbers(int a, int b)
27     {
28         int result = a + b;
29         System.out.println("Two numbers added together : " + result);
30     }
31 }
```

In this example the method addNumbers takes two parameters in its actual parameter list (between these). This is consistent with the way it was declared in the method declaration below.

This is the method signature formal parameter list. Notice that we must state the type of each of the formal parameters and that they are separted by commas. When we use this method we must supply it with two int parameters otherwise an compile error wil occur.

The scope of the formal parameter a and b as well as the local variable result is only within this method. After the method has finished running these variable no longer exist. a and b hold copies of the contents of the formal parameters num1 and num2

(3) Returns an int and takes two parameters

```
1  /**
2   * This program illustrates the essential aspects of declaring a method
3   * and using this method in the main method of the application.
4   *
5   * @author Joe Duffin
6   * @version 4/09/2013
7   */
8  public class MethodSimple3
9  {
10     public static void main(String args[])
11     {
12         int num1 = 5;
13         int num2 = 6;
14         int num3 =0;
15         // This is a method call or a method invocation.
16         // Notice that this method is designed to take two int parameters
17         // and also to return and int value (send back to where it was called)
18         num3 = addNumbers(num1, num2); // this is a method call or a method invocation.
19
20         System.out.println("The sum of " + num1 + " and " + num2 + " is : " + num3);
21     }
22     /**
23      * This method takes two integers, adds them together and
24      * prints the result to the screen
25      */
26     // The first line below is the method signature.
27     // the key word int before the method name means that it returns a value of type int.
28     // Notice also that the method is designed to take two parameters a and b
29     public static int addNumbers(int a, int b)
30     {
31         int result = a + b
32         return result; // if the method has a return type then it must have the word return in it.
33     }
34 }
```

This is a call to the method addNumber supplying two actual parameters, num1 and num2. This method has been designed to return and int value and that is why we have num3 assigned to take the returned value. The use of addNumbers here is consitent with the way it is declared below. Notice the method signature in the method declaration.

method call

The method signature of this method declaration tells us that the method takes two int parameters and returns an int value. As we said before we must use the method according to its signature.

method signature

(4) Passing an array reference to a method

```
9  public class SimpleArraysMethodPassing
10 {
11     public int value = 0;
12
13     public static void main(String args[])
14     {
15
16         // myNumbers stores the machine address of an integer array which is created in memory
17         // and intialised in sequence with the values below
18         int[] myNumbers = {51,62,3,7,9,15,27,29,2,0,10};
19         // the loop below prints out each element in the myNumbers array.
20         for(int i= 0;i< myNumbers.length; i++)
21         {
22             System.out.println("Array element: [" + i +"] contains : " + myNumbers[i]);
23         }
24
25         changeContents(myNumbers); //call this method to begin executing at this point in the program.
26
27         // the loop below prints out each element in the myNumbers array.
28         for(int i= 0;i< myNumbers.length; i++)
29         {
30             System.out.println("Array element: [" + i +"] contains : " + myNumbers[i]);
31         }
32
33     }
34     // This method will change the contents of the array referred to by the variable myList
35     public static void changeContents(int [] myList)
36     {
37         // set each and every array element to 99.
38         for (int i=0;i<myList.length;i++)
39         {
40             myList[i] = 99;
41         }
42     }
43 }
```

Call the method changeContents passing it the array variable.

This array variable myNumber contains a link or reference to the start of the myNumbers array. We don't actually pass the whole series of memory boxes to the method, just the link to

This second for loop prints out the contents of the array again to show that that all the memory units have had 99 stored in them.

void before the method name tell us that the method does not return a value.

This is the formal parameter list of the method. This method takes one paramet which is an array variable. When we use this method we must make sure that it only receives one variable of type int [].

(5) Returning a newly created array reference.

```
8   */
9   public class SimpleMethodReturingArray
10  {
11      public int value = 0;
12
13      public static void main(String args[])
14      {
15
16          int [] myList; // This is an array variable (it is capable of storing a link to an array)
17
18          myList = createNewList(); //Call the method createNewList which returns a link to a new array
19
20          for(int i= 0;i< myList.length; i++) // print out the contents of the array.
21          {
22              System.out.println("Array element: [" + i +"] contains : " + myList[i]);
23          }
24
25      }
26      // This method will create an array, initialise all the values to 28 and return
27      // a reference to the array to the main method.
28      public static int [] createNewList()
29      {
30          int [] boxes = new int[10];
31
32          // set each and every array element to 99.
33          for (int i=0;i<boxes.length;i++)
34          {
35              boxes[i] = 28;
36          }
37          return boxes; // returns a link (reference) to the newly created and initialised array.
38
39      }
40  }
41
```

Call the method createNewList. Notice that it does not take any parameters but it returns an reference (link) to an int array.

This line asks the computer for an array of 10 ints and it stores a link to this array in the array variable boxes.

Use a for loop to fill each and every array element with the number 28

Return the array reference stored in boxes to point at from where the method was called

Rules for writing and using methods.

1. The use or call of a method must match the signature declaration of the method.

2. If the method returns a value of a particular type then the receiving variable should be capable of storing that type.

3. The number and order of the parameters in the method call (actual parameter list) must match the number and order of parameters in the method signature declaration (formal parameter list).

4. The type of the parameters in the both the method call and the method declaration must match.