Class Scanner

Class Scanner is part of the package **java.util .** It allows us to read input from the keyboard and store it into a variable inside our program, which we can use in later calculations. User input helps to make our programs more interesting.

To read input, you must do the following steps:

1. Declare an **instance** of class scanner, to be your "**input stream**" through which your program will receive data:
Scanner **identifier** = new Scanner(System.in);

Ie. Scanner input = new Scanner(System.in);

System.in is the "standard input stream" or a reference to the keyboard. Once created a Scanner object can be used throughout your program.

2. For each data item the program needs to read, a prompt must be displayed to the user.
System.out.println("Please enter your name and age");
This is important… if you want input from your user you need to tell them exactly what you need.

3. Read the value input by the user using one of the methods for class **Scanner** (See the JAVA API for descriptions of the methods) the methods include:

next() - reads any sequence of characters until either a blank space, or a "the end of line" is encountered. This method returns a "String" value. A String is a special type of object that contains any sequence of characters (letters, special characters, digits and even blank spaces)
nextLine() - Reads all characters until the end of the current input line. This method returns a value of type String.
nextInt()- This reads any sequence of digits (including a + or -), and produces an integer value. The end of the number is indicated by either the end of the line, a blank space, or any non-digit character.
nextDouble()- This reads any sequence of digits, a sign, and decimal point and produces a value of type double. The end of the number is indicated by a blank space, the end of line, or any other non-digit character.

There is a method from class scanner to READ input into any type of variable. There is one "next" method for each of the primitive data types and **next & nextLine** for strings. NOTE in the definition for class scanner you will notes that the keyword "static" is absent from the description of Scanner's methods. This means that the methods must be applied to an "instance" of a Scanner so that the system knows from where to gather the input.

For example to read an integer from the keyboard:

```
int num;
System.out.println("Please input an integer number.");
num = input.nextInt();
```

How is input viewed?

When a user enters values via the standard input device a user can enter one or more values on one line before pressing enter. The value entered is viewed as a series of individual characters that are reformatted by the Scanner methods into more complex values such as Strings, ints, and floats. So for example, if the user receives the following prompt:

Please enter your 3 exam scores in the range from 0 to 100 followed by your name:

The user can respond by entering everything on one line:

95.5 100 75 John Doe

Or entering values one per line pressing enter after each value:
95.5
100
75
John Doe

Pressing enter after each individual value.

If we prompt the user for multiple values, inside the program we will need multiple input statements to read the values into variables:

So Assuming we have the following variables declared:

```
double exam1, exam2, exam 3:
String name;
Scanner   input = new Scanner(System.in);
```

and display the following prompt:

```
System.out.println("Please enter your 3 exam scores in the range from 0 to 100 " +
        "followed by your name:  ");
```

We would need the following statements to actually read the data typed by the user into our variables:

exam1 = input.nextDouble();
exam2 = input.nextDouble();
exam3 = input.nextDouble();
name = input.nextLine();

Assuming the user typed:

95.5 100 75 John Doe

Here is how the input statements would be executed:

1. **exam1 = input.nextDouble();** --- This would take characters from the input stream that were either digits or decimal points and would stop at the first blank space or a carriage return. This method would extract the characters **95.5** from the input stream, translate them into a numeric value of type double and store them in the variable exam1.

2. **exam2 = input.nextDouble();** ----- This method would extract the characters **100** from the input stream, translate them into a numeric value of type double and store them in the variable exam2.
3. **exam3 = input.nextDouble();** ---- This method would extract the characters **75** from the input stream, translate them into a numeric value of type double and store them in the variable exam3.
4. **name = input.nextLine();** --- This method will take ANY sequence of characters, blanks, digits and special characters until the end of the line and stores them as a character string into the specified variable. It will place **"John Doe"** into the variable name

   NOTE: with the exception of nextLine() the methods remove any blanks from the input.

What would happen if :

What would happen if you tried to read the wrong type of value from the keyboard or if the user typed the wrong type of information?

For example if the user typed 95.5 and the input was read via:

exam1 = input.nextInt();

When reading a number if the Scanner encounters a character then the error "**`java.util.InputMismatchException`**" would be generated, and execution would stop. This is an example of a runtime exception. Runtime exceptions are errors that occur while the program is executing. This error would also happen if we are trying to read in an integer type number and instead the user types either a letter or a floating point number containing a decimal point.

Example: #2 Another example using both Scanner and Math

Write a program that prompts the user to input a decimal number and outputs the number rounded to the nearest integer.

```
import java.util.*;

public class readDecimal {

   public static void main(String args[]) {


        Scanner  input = new Scanner(System.in);

        double decimalNumber;
        long     intNumber;

        System.out.println("Please enter a decimal number: ");
        decimalNumber = input.nextDouble();

        intNumber = Math.round(decimalNumber);

        System.out.println(decimalNumber +
                " rounded to the nearest integer is " + intNumber);

}
}
```

Sometimes a problem with Scanner:

When a user enters values, they may enter values either one per line, or multiple values on the same line.

```
import java.util.*;

public class readValues {

  public static void main(String args[]) {


        Scanner  console = new Scanner(System.in);

        float gpa;
        short   age;
        String fname, lname;
        String streetAddress;

        System.out.println("Please enter your first and last name on the same line");
         fname = console.next();
         lname = console.nextLine();

        System.out.println("Please enter your gpa, and age on the same line.");

        gpa = console.nextFloat();
        age =  console.nextShort();

        System.out.println("The students name is : " + fname + " " + lname +
        " with gpa " + gpa  + " and age " + age);
}
}
```

Valid input would be:

John Jones
3.2 21

Problems with combining calls to nextLine and other Scanner functions.

Problems can arise when a program mixes calls to nextLine, with calls to other scanner functions, especially those such as nextInt and nextDouble. There is a slight discrepancy in how these methods operate, which can result in input seemingly being skipped.

For example, consider the following code:

```
import java.util.Scanner;

/* this program has a problem reading input */

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

    Scanner input = new Scanner(System.in);

    int age;
    double income;
     String name;

     System.out.println("What is your age? ");
     age = input.nextInt();

     System.out.println("What is your income?");
     income = input.nextDouble();

     System.out.println("What is your full name?");
      name = input.nextLine();

      System.out.println("Hello!!! My name is " + name + "I am "+ age+ " years old, "
                  + "and earn " + income + " per year!!!" );
}
}
```

I would like to execute this program using the data John Doe, 23, and 23456.00. However when I execute the program , I get the output:

```
What is your age?
25
What is your income?
65000
What is your full name?
```

Hello!!! My name is I am 25 years old, and earn 65000.0 per year!!!

It appears that the input statement to read an individual's name is skipped.
(name = input.nextLine();)

What happened???? Class Scanner has some slight inconstancies about how it "next" methods are implemented.

When data is input from the keyboard, it is temporarily stored in a location in memory called the "input buffer" until a method is executed to read that data (**data stays in the input buffer until it is read-)**. When the read happens, data is moved from the input buffer into a variable. For example when prompted for a persons age, when 23 is typed in, what is really stored into the input buffer is :

23\r

where "\r" represents the carriage return, generated by pressing the enter key.
Methods such as next(), nextInt(), nextDouble() and so on, do NOT remove the carriage return from the input buffer. So in our example when the persons age is read from the input buffer using nextInt() it removes the 2 & 3, creates the integer 23 and returns that value. The "\r" is left in the input buffer. When the user enters the next value, the input buffer contains:

\r23456.00\r

the call to nextDouble() to read a person's income, first skips the beginning "\r", reads 23456.00 and returns it, but leaves the buffer as:

\r

when nextLine() is executed to read an individual's name, it reads characters until a "\r" is encountered. It does not skip "leading" carriage returns. So the method immediately exits without allowing the user to input additional values because it already found the end of the line.

The program can be easily fixed by adding an extra invocation of nextLine() after reading the two numeric values to consume the extra "\r".

```
import java.util.Scanner;

/* this program has a problem reading input */

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

    Scanner input = new Scanner(System.in);

    int age;
    double income;
     String name;

     System.out.println("What is your age? ");
     age = input.nextInt();

     System.out.println("What is your income?");
     income = input.nextDouble();

     //dummy input statement removes the extra newline character
     input.nextLine();

     System.out.println("What is your full name?");
      name = input.nextLine();

      System.out.println("Hello!!! My name is " + name + "I am "+ age+ " years old, "
                  + "and earn " + income + " per year!!!" );
}
}
```