# The Scanner Class

- The `Scanner` class provides convenient methods for reading input values of various types

- A `Scanner` object can be set up to read input from various sources, including the user typing values on the keyboard

- Keyboard input is represented by the `System.in` object

# Reading Input

- The following line creates a `Scanner` object that reads from the keyboard

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

- The `new` operator creates the `Scanner` object

- Once created, the `Scanner` object can be used to invoke various input methods, such as

```
answer = scan.nextLine();
```

# Reading Input

- The `Scanner` class is part of the `java.util` class library, and must be imported into a program to be used

- The `nextLine` method reads all of the input until the end of the line is found

- We'll discuss the details of object creation and class libraries later

- Some methods of the `Scanner` class:

```
Scanner (InputStream source)
Scanner (File source)
Scanner (String source)
        Constructors: sets up the new scanner to scan values from the specified source.

String next()
        Returns the next input token as a character string.

String nextLine()
        Returns all input remaining on the current line as a character string.

boolean nextBoolean()
byte nextByte()
double nextDouble()
float nextFloat()
int nextInt()
long nextLong()
short nextShort()
        Returns the next input token as the indicated type. Throws
        InputMismatchException if the next token is inconsistent with the type.

boolean hasNext()
        Returns true if the scanner has another token in its input.

Scanner useDelimiter (String pattern)
Scanner useDelimiter (Pattern pattern)
        Sets the scanner's delimiting pattern.

Pattern delimiter()
        Returns the pattern the scanner is currently using to match delimiters.

String findInLine (String pattern)
String findInLine (Pattern pattern)
        Attempts to find the next occurrence of the specified pattern, ignoring delimiters.
```

```java
//************************************************************
//  Echo.java        Java Foundations
//
//  Demonstrates the use of the nextLine method of the Scanner class
//  to read a string from the user.
//************************************************************

import java.util.Scanner;

public class Echo
{
    //--------------------------------------------------------------
    //  Reads a character string from the user and prints it.
    //--------------------------------------------------------------
    public static void main(String[] args)
    {
        String message;
        Scanner scan = new Scanner(System.in);

        System.out.println("Enter a line of text:");

        message = scan.nextLine();

        System.out.println("You entered: \"" + message + "\"");
    }
}
```

# Input Tokens

- Unless specified otherwise, *white space* is used to separate the elements (called *tokens*) of the input

- White space includes space characters, tabs, new line characters

- The `next` method of the `Scanner` class reads the next input token and returns it as a string

- Methods such as `nextInt` and `nextDouble` read data of particular types

```java
//************************************************************************
//  GasMileage.java       Java Foundations
//
//  Demonstrates the use of the Scanner class to read numeric data.
//************************************************************************

import java.util.Scanner;

public class GasMileage
{
   //---------------------------------------------------------------------
   //  Calculates fuel efficiency based on values entered by the
   //  user.
   //---------------------------------------------------------------------
   public static void main(String[] args)
   {
      int miles;
      double gallons, mpg;

      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the number of miles: ");
      miles = scan.nextInt();

      System.out.print("Enter the gallons of fuel used: ");
      gallons = scan.nextDouble();

      mpg = miles / gallons;

      System.out.println("Miles Per Gallon: " + mpg);
   }
}
```