جامـــــعـة المــــسـتـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قـســــــــــــم الانظمة الطبية الذكية

Lecture: (7)

Input and output Basics Part I

Subject: Computer Programming (I)
Level: First
Lecturer: Dr. Maytham N. Meqdad

The print() and println() Methods

Print Text

You learned from the previous chapter that you can use the `println()` method to output values or print text in Java:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

You can add as many `println()` methods as you want. Note that it will add a new line for each method:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World!");
    System.out.println("I am learning Java.");
    System.out.println("It is awesome!");
  }
}
```

Double Quotes

When you are working with text, it must be wrapped inside double quotations marks `""`.

If you forget the double quotes, an error occurs:

```
public class Main {
  public static void main(String[] args) {
    System.out.println(This sentence will produce an error);
  }
}
-----------------------------------------------------------------------
public class Main {
  public static void main(String[] args) {
    System.out.println("This sentence will work!");
  }
}
```

The Print() Method

There is also a `print()` method, which is similar to `println()`.

The only difference is that it does not insert a new line at the end of the output:

```
public class Main {
  public static void main(String[] args) {
    System.out.print("Hello World! ");
    System.out.print("I will print on the same line.");
  }
}
```

▪ Java User Input (Scanner)

The Java Scanner class is a simple, versatile, easy-to-use class that makes user input in Java relatively straightforward.

To perform user input with the Scanner class, follow these steps:

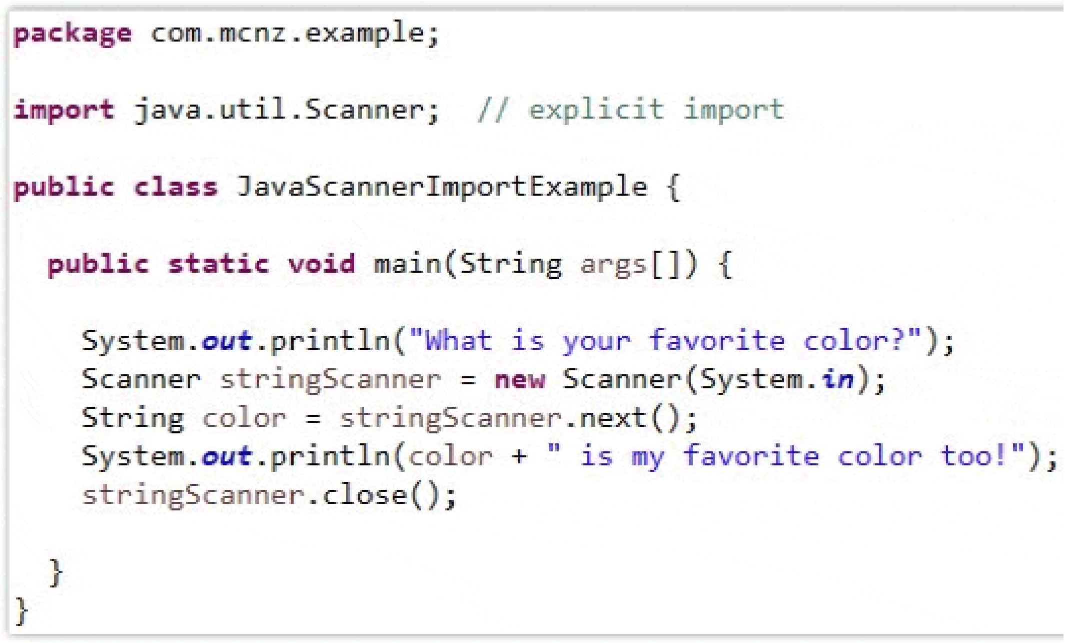
1. Create an instance of the Scanner with the new keyword.
2. Specify the System.in as the argument for the Scanner constructor.
3. Optionally set a delimiter other than the enter key.
4. Use the Scanner's `next ()` or `nextLine()` methods to convert user input into the appropriate type.
5. Use the Java user input in your program.

ava Scanner import example

Found in the java.util package, Java's Scanner class can read input from the command line and return it as a String, BigDecimal or any one of Java's eight primitive types.

To use the Java Scanner class, you must either:

- import `java.util.Scanner`
- import `java.util.*;`
- reference the package and class name `java.util.Scanner` in your code



```java
package com.mcnz.example;

import java.util.Scanner;  // explicit import

public class JavaScannerImportExample {

  public static void main(String args[]) {

    System.out.println("What is your favorite color?");
    Scanner stringScanner = new Scanner(System.in);
    String color = stringScanner.next();
    System.out.println(color + " is my favorite color too!");
    stringScanner.close();

  }
}
```

For the most part, Java's Scanner class is fairly easy to use, as the methods are largely self-explanatory.

Examples of straightforward Scanner methods to get data such as floats, doubles or text include:

- `nextInt()`
- `nextByte()`
- `nextLong()`
- `nextFloat()`
- `nextDouble()`
- `nextLine()`
- and just plain `next()`

Java Scanner String example

One thing I don't like about Java's Scanner class, especially when it's taught to novice Java programmers, is that it introduces several advanced concepts. Specifically, these are:

- Import statements — the java.util package must be imported.
- The *new* keyword — an instance of the Scanner must be instantiated.

If those two advanced topics don't intimidate the developer, then user input with Java's Scanner class is a lead-pipe cinch.

```java
import java.util.Scanner;
public class ScannerUserInput {
  public static void main(String[] args) {
    // Java Scanner String input example
    System.out.println("What is your name?");
    Scanner scanner = new Scanner(System.in);
    String name = scanner.nextLine();
    System.out.println(name + " is a nice name!");

  }
}
```

In the example above of how to use Java's Scanner for user input, the import statement is at the start of the code, along with the creation of an instance of the Scanner with the *new* keyword. The import can optionally be removed if Java's Scanner class explicitly references the package.

Java Scanner int example

The following example of Java user input with the Scanner class avoids the use of an import statement:

```java
// Java Scanner int input example
System.out.println("What is your age?");
java.util.Scanner scanner = new java.util.Scanner(System.in);
int age = scanner.nextInt();
System.out.println("I remember being " + age + " years old!" );

// Java Scanner String input example
System.out.println("Where were you born?");
String city = scanner.nextLine();
System.out.println("I hope to visit " + city + " some day." );
```

This example also demonstrates how to use one instance of the Scanner class multiple times within the same program. You do not need to instantiate it with the new keyword each time.

Java Scanner hasNext() example

To continually grab input from the user, you can use the Scanner's `hasNext()` method as the condition for a while loop. This will cause the program to continually take input from the user until the program either shuts down, or encounters a break statement.

Here's a Scanner `hasNext()` example that adds numbers until the total sum exceeds 100:

```
System.out.println("Enter some numbers to add: ");
Scanner scanner = new Scanner(System.in);
int count = 0;
while (scanner.hasNext()) {
  count = count + scanner.nextInt();
  System.out.println(count);
  if (count > 100) {
    System.out.println("Max exceeded!");
    break;
  }
}
```

Java Scanner delimiter example

By default, the scanner uses the enter key to indicate the user has finished their user input. But this can be changed by through the use of the `useDelimiter()` method.

The following Scanner example takes a String of comma-separated values (CSVs) and prints them out one at a time. The program passes the text String to the Scanner's constructor, and then changes the delimiter to a comma.

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    String csv = "a,b,c,d,e";
    Scanner scanner = new Scanner(csv);
    scanner.useDelimiter(",");
    while (scanner.hasNext()) {
      System.out.println(scanner.next());
    }
  }
}
```

When this Java Scanner example runs, the output is:

```
a
b
c
d
e
```

Notice that the program ignores the commas in the original text String.

Java Scanner char input example

Interestingly, the Java Scanner char input is not supported through a defined method in the Scanner class.

However, it is possible to have a Scanner input one char at a time through the use of the delimiter setting and the Scanner's `hasNext()` method.

The following example takes char input with the Scanner:

```
import java.util.Scanner;
public class NextCharScanner{

  // Java Scanner char input example
  public static void main(String[] args) {
    System.out.println("Provide the Java Scanner char input: ");
    Scanner charScanner = new Scanner(System.in);
    charScanner.useDelimiter("");
    while (charScanner.hasNext()) {
      char name = charScanner.next().charAt(0);
      if (name == '\n') {
        return;
      }
    }
  }
}
```