In this section, we will cover using the Scanner class to access the user’s input via the keyboard.

Java Scanner

CSCI 1250 Study Guide

Schneider, Michael Joseph

Scanner Object

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

    public static void main(String[] args)
    {
        Scanner kb = new Scanner(System.in);
    }
}
```

Import Statement

An import statement tells the compiler that there is code outside the file that is being referenced. The compiler will find that code and "pull" it in for compiling.

java.util – Package name

Scanner – Name of class being used.

Name of Scanner Object

Just as a String object is given a name to reference its data, a Scanner object must be given a name to reference its data & methods. Any name can be used, common names are kb, scnr, and keyboard.

Initialize Object

To create an object, Java needs to know all the data the object's Class needs. To do this it uses:

ClassName var = `new ClassName(parameters);`

new – place the object's data in memory and return the data's address(location) in memory.

ClassName – name of the class being used (actually a Constructor for the Class, but that will be covered later).

(parameters) – Any initial data the object requires.

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

    Scanner - Call the Scanner class's
    constructor.

    System.in - reference the default input
    stream (a.k.a. the keyboard).
```

Example Scanner Code

```
import java.util.Scanner;

public class InputExample
{
    public static void main(String[] args)
    {
        Scanner kb = new Scanner(System.in);
        System.out.print("Name:  ");
        String name = kb.nextLine();
    }
}
```

Method Call

To grab input from the Scanner object, you will have to write the name of your Scanner, kb, and the method to grab your data, .nextLine().

Scanner contains a specific method to grab each data type. A full list of it's methods can be found in the Scanner JavaDoc : https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

Data Type Returned	Method Name	Example
String	.nextLine()	String name = kb.nextLine();
int	.nextInt()	int number = kb.nextInt();++
double	.nextDouble()	double decimal = kb.nextDouble();
boolean	.nextBoolean()	boolean check = kb.nextBoolean;

Possible errors

The Scanner methods will only grab their data type from the keyboard. If the user enters a bad value, your program will crash. Example:

double decimal = kb.nextDouble();//User enters "Hello" , the program will crash!

Later on we will cover how to validate user input, for now be aware of this potential problem.

++.nextInt() can be difficult to use because it will only take the whole numbers from the keyboard and leave behind the carriage return (enter key). This means if we call .nextLine() after .nextInt(), it will only return the remaining carriage return and "skip" the user's actual input.

Uncleared Buffer Example

```
import java.util.Scanner;

public class InputExample
{
    public static void main(String[] args)
    {
        Scanner kb = new Scanner(System.in);

        System.out.println("Please enter a whole number: ");

        int number = kb.nextInt();//Only "grabs" the whole number
                                  //Leaves behind the carriage return

        System.out.print("\nName: ");

        String name = kb.nextLine();//Grabs the remaining carriage return!
    }
}
```

Cleared Buffer Example

```
import java.util.Scanner;

public class InputExample
{
    public static void main(String[] args)
    {
        Scanner kb = new Scanner(System.in);

        System.out.println("Please enter a whole number: ");

        int number = kb.nextInt();//Only "grabs" the whole number
                                  //Leaves behind the carriage return

        kb.nextLine();//This call to .nextLine() will remove the carriage
                      //return

        System.out.print("\nName: ");

        String name = kb.nextLine();//Now we can accept the user's input
    }
}
```