Set 6: Input From the Keyboard

Skill 6.1 Import the java.io and java.util classes

Skill 6.1 Concepts

In addition to the *Math* and *String* classes we learned about earlier, Java has many more classes we can access. But, unlike the *Math* and *String* classes, because they are not included with the core, they must be imported.

The class we are going to learn about today is the *Scanner* class. The Scanner class allows us to get input from the user whether it be from a file or the keyboard. The following code illustrates how to import the libraries required to get user input using the Scanner class.

These must go at the top of the class, before the class declaration.

```
import java.io.*;
import java.util.*;
public class ScannerPractice{
}
```

With the above libraries imported, we can access methods that will make our Java programs much more interactive!

Skill 6.1 Exercise 1

Skill 6.2 Use the Scanner class to create a keyboard reader object

Skill 6.2 Concepts

Before we can access the methods available to us, we must create a *Scanner object*. You can call the Scanner object anything you like, so long as it follows the Java variable naming conventions. The code below creates a Scanner object called *kbReader*. What each part of the code represents is broken down below,

- **Scanner:** The type of variable
- **kbReader:** The name of the variable
- **new:** Tells Java we are creating a Scanner object
- **Scanner(System.in):** The type of Scanner object we are creating. In this case we will be reading input from the keyboard - System.in

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

Skill 6.2 Exercise 1

Skill 6.3 Retrieve and display user input of type int, double, and String

Skill 6.3 Concepts

The *kbReader* object we created above will allow us to get user input. This process is illustrated below.

User input of type int

Getting int data types from the user requires the *nextInt()* method. Consider the following code,

Code	Output

`Scanner kbReader = new Scanner(System.in);`
`System.out.println("How old are you?");`
`int age = kbReader.nextInt();`
`String result = "You were born in " + (2025 - age);`
`System.out.println(result);`
	How old are you?
	16 ← user input
	You were born in 2009

In the above code, the line `int age = kbReader.nextInt();` is doing two things: (1) it is prompting the user for an integer input from the keyboard and (2) it is assigning the user input to the int type variable *age*.

Also notice the *dot* notation. We have used this before. The *dot* notation allows us to access the Scanner libraries we imported previously.

User input of type double

Getting double data types from the user requires the *nextDouble()* method. Consider the following code

Code

`Scanner kbReader = new Scanner(System.in);`
`System.out.println("How tall are you in inches to the nearest 10th decimal place?");`
`double height = kbReader.nextDouble();`
`String result = "You are " + (int)(height/12) + " feet " + (height % 12) + " inches";`
`System.out.println(result);`

Output
How all are you in inches to the nearest 10th decimal place? 63.5 ← user input You are 5 feet 3.5 inches

In the above code, the line `double height = kbReader.nextDouble();` is doing two things: (1) it is prompting the user for a double input from the keyboard and (2) it is assigning the user input to the double type variable *height*.

User input of type String

The *next()* method accepts String input up to the first white space. For example if I type *Computer Science* as input, the *next()* method only scans the word "Computer". This is illustrated below.

Code

```
Scanner kbReader = new Scanner(System.in);
System.out.println("What is the name of this class?");
String firstWord = kbReader.next();
System.out.println(firstWord);
```

Output

```
What is the name of this class
Computer Science <-user input
Computer
```

The *nextLine()* method can be used to retrieve an entire line, including white space. This is illustrated below.

Code

```
Scanner kbReader = new Scanner(System.in);
System.out.println("What is the name of this class?");
String firstWord = kbReader.nextLine();
System.out.println(firstWord);
```

Output

```
What is the name of this class
Computer Science <-user input
Computer Science
```

Skill 6.3 Exercises 1 thru 3

Skill 6.4 Beware of the Scanner anomaly

Skill 6.4 Concepts

In Java, when you use *nextInt(), nextDouble*, or *next(),* it reads only the value that's entered, but does not consume the newline character (\n) left in the input buffer after you press Enter.

If you immediately call *nextLine()* after *nextInt(), nextDouble*, or *next()*, it reads this leftover newline as an empty string, not the next line of actual input.

See the result below,

Code

```
Scanner kbReader = new Scanner(System.in);
System.out.println("How old are you?");
int age = kbReader.nextInt();
System.out.println("What is the name of this class?");
String myClass = kbReader.nextLine();
System.out.println(myClass);
```

Output

```
How old are you?
16 <-user input
What is the name of this class?
                <-reads \n
PS C:\Users\PLUSKH01\Desktop\ForDDrive>
```

To fix this, add an extra *nextLine()* after *nextInt(), nextDouble*, or *next()* to consume the newline,

Code

```
Scanner kbReader = new Scanner(System.in);
System.out.println("How old are you?");
int age = kbReader.nextInt();
kbReader.nextLine(); <-Consumes \n
System.out.println("What is the name of this class?");
String myClass = kbReader.nextLine();
System.out.println(myWord);
```

Output

```
How old are you?
16 <-user input
What is the name of this class?
Computer Science <-user input
Computer Science
```