Tutorial: Java Scanner Class and Methods

Goals:

- Understand the functionality of the **Scanner** class
- Learn how to use the **Scanner methods**

Description:

You have been working with a **JOptionPane** on your first two labs. More specifically, you have used the **input dialog** in order for the user to input something into the popup. What we will now learn to use is a **Scanner** object. It has a similar idea as an **input dialog** in the sense that it takes in user input, but instead, you enter input into your console. In this tutorial, we will show you the basics of the **Scanner** Object.

Contents:

Demo:

1. Here we will create a class called ScannerDemo.java:

ScannerDemo.java

```
/**
  * CS180 - Scanner Tutorial
  * Explain briefly the functionality of the program
*/

public class ScannerDemo {

    public static void main(String[] args) {

    }

}
```

2. What we want to start with is to initialize the Scanner object. We do this with the following line of code:
   a. **Scanner scan = new Scanner(System.in);**
      i. The **System.in** parameter allows us to take in user input.
3. We want to add two integers together. To do this we need to ask the user to type in two numbers.
   a. We will print the question by simply using **println();**
4. Then after that, we want to finally create an integer with the input that was given. We will write this line of code:
   a. **int a = scan.nextInt();**

5. Repeat the process above so we have two inputs and we should now have the following:

ScannerDemo.java

```
public class ScannerDemo {

    public static void main(String[] args) {

        Scanner scan = new Scanner();

        System.out.println("Please type in the first number: ");
        int a = scan.nextInt();

        System.out.println("Please type in the second number: ");
        int b = scan.nextInt();

        System.out.println(a+b);

        scan.close();

    }

}
```

next() Methods:

1. When we are using the **Scanner**, we always you a **next()** method. Scanner has many of these methods, for all **primitive** types. Here's a small list of relevant ones you need to know:

Scanner Methods
nextBoolean()
nextByte()
nextDouble()
nextFloat()
nextInt()
nextLine()
nextShort()
next()

2. All of these return to you a respected primitive type. **nextLine()** is used to read a **String**, and **next()** is used to read any general pattern.

hasNext() Methods:

1. Not only can we use this for input, but we can utilize a scanner on really any String.
2. Let's go back to out ScannerDemo.java, and we will add a method called **parseInt().**
3.

ScannerDemo.java

```
public int parseInt(String numbers) {

}
```

4. Our goal with this method is that we assume that the String is a list of numbers separated by spaces, ex: "1 2 3 4 5".
5. What we want to do is to take these numbers and add them together.
6. We can do this by using methods called **hasNextPrimitive()**.
    a. The primitive is referring to any primitive type.
    b. We will start by creating our **Scanner**, but instead of passing **System.in** as a parameter, we will pass in the String **numbers**:
        i. **Scanner scan = new Scanner(numbers);**

Demo 2:

1. Since we are dealing with **integers**, we will use the method **hasNextInt().** All of these **hasNext()** methods return Boolean types.

2. What we can do is we can have a while loop going.
3. This is how this looks in code:

ScannerDemo.java

```
public int parseInt(String numbers) {
    Scanner scan = new Scanner(numbers);
    while(scan.hasNextInt()) {
        //implementation
    }
}
```

4. Now before the while loop, we want to set **two integers: sum and reference**. Sum is the total sum, and reference will be the number we are referencing in the list.
5. We will now add:
    a. **reference = scan.nextInt();**
6. What we want to the while loop is to add reference to the sum. This is done this way:

ScannerDemo.java

```
public int parseInt(String numbers) {
     int sum = 0;
     int reference;


     Scanner scan = new Scanner(numbers);
     while(scan.hasNextInt()) {
         reference = scan.nextInt();
         sum += reference;
     }

     return sum;
}
```

Appendix:

Later on, you will learn about catching **Exceptions** in a **try-catch** block. Examples of **Exceptions** in a **Scanner** could include:

1. The user not inputting anything
2. The user inputting the wrong primitive type.
3. Going out of bounds on a String

Go back to the **ScannerDemo** main method:

ScannerDemo.java

```java
public class ScannerDemo {

    public static void main(String[] args) {

        Scanner scan = new Scanner();

        System.out.println("Please type in the first number: ");
        int a = scan.nextInt();

        System.out.println("Please type in the second number: ");
        int b = scan.nextInt();

        System.out.println(a+b);

        scan.close();

    }

}
```

1. Say that someone inputs letters when asked to input a number. We will get an **Exception** because a number wasn't inputted. The program will fail and we are left without a solution.
2. Refer to the **Exceptions** tutorial when instructed to do so and you will learn how to handle these **Exceptions** with statements.
3. Then you can come back here and fix the problem of a possible **Exception**. We want the person to be **prompted** to input an **integer** if one was not initially inputted.

If you have any questions, please visit TA office hours, or ask a question on Piazza.