Scanner hasNext Methods

Java Scanner API

Recap – next Methods

Return	Method	Description
String	next()	Finds and returns the next complete token from this scanner.
String	nextLine()	Advances this scanner past the current line and returns the input that was skipped.
int	nextInt()	Scans the next token of the input as an int.
double	nextDouble()	Scans the next token of the input as a double

Error Handling

- `InputMismatchException`
  - If the next token does not match the pattern for the expected type, or is out of range for the expected type

```java
import java.util.Scanner;

public class CheckRaceResults {
    public static void main (String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter place (int): ");
        int place = input.nextInt();

        if (place <= 3) {
            System.out.println("You earned a medal!");
        } else {
            System.out.println("Finisher!");
        }
    }
}
```

```
$ javac CheckRaceResults.java
$ java CheckRaceResults
Enter place (int): 1
You earned a medal!

$ java CheckRaceResults
Enter place (int): one
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at CheckRaceResults.main(CheckRaceResults.java:9)
```

Error Handling

- `InputMismatchException`
  - If the next token does not match the pattern for the expected type, or is out of range for the expected type
- `NoSuchElementException`
  - If the input is exhausted

hasNext Methods

next methods – Actual READ in
hasNext methods – CHECK only

Return	Method	Description
String	next()	Finds and returns the next complete token from this scanner.
boolean	**hasNext()**	**Returns true if this scanner has another token in its input.**
String	nextLine()	Advances this scanner past the current line and returns the input that was skipped.
boolean	**hasNextLine()**	**Returns true if there is another line in the input of this scanner.**
int	nextInt()	Scans the next token of the input as an int.
boolean	**hasNextInt()**	**Returns true if the next token in this scanner's input can be interpreted as an int value using the nextInt() method.**
double	nextDouble()	Scans the next token of the input as a double
boolean	**hasNextDouble()**	**Returns true if the next token in this scanner's input can be interpreted as a double value using the nextDouble() method.**

Q: What's the output given the input as ➡️

```
$ javac HasNextMethods.java
$ java HasNextMethods
Token(s): CS1101
hasNextInt = false
hasNextDouble = false
hasNext = true
hasNextLine = true

$ java HasNextMethods
Token(s): CS 1101
hasNextInt = false
hasNextDouble = false
hasNext = true
hasNextLine = true
```

```
$ java HasNextMethods
Token(s): 1101
hasNextInt = true
hasNextDouble = true
hasNext = true
hasNextLine = true

$ java HasNextMethods
Token(s): 110.1
hasNextInt = false
hasNextDouble = true
hasNext = true
hasNextLine = true
```

```java
import java.util.Scanner;

public class HasNextMethods {
    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Token(s): ");

        System.out.println("hasNextInt = " + input.hasNextInt());
        System.out.println("hasNextDouble = " + input.hasNextDouble());
        System.out.println("hasNext = " + input.hasNext());
        System.out.println("hasNextLine = " + input.hasNextLine());
    }
}
```

hasNext methods DO NOT consume input

Robust Programs!

- Robustness
  - The degree to which erroneous situations are handled gracefully
- Want to write programs that execute when we present illegal data
  - Testing provides the illegal data
  - Now want to handle it

Handling User Errors

```java
import java.util.Scanner;

public class CheckRaceResults {
    public static void main (String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter place (int): ");

        // Check if next token can be read in as an int
        while (!input.hasNextInt()) {
            // discard the invalid token by reading in the token
               but not assigning it to any variable
            input.next();
            System.out.println("Not an int, try again.");
            System.out.print("Enter an int: ");
        }
        // ASSERT: the next token can be read as an int
        int place = input.nextInt();

        if (place <= 3) {
            System.out.println("You earned a medal!");
        } else {
            System.out.println("Finisher!");
        }
    }
}
```

```
$ javac CheckRaceResults.java
$ java CheckRaceResults
Enter place (int): one
Not an int, try again.
Enter an int: 1one
Not an int, try again.
Enter an int: 1
You earned a medal!
```