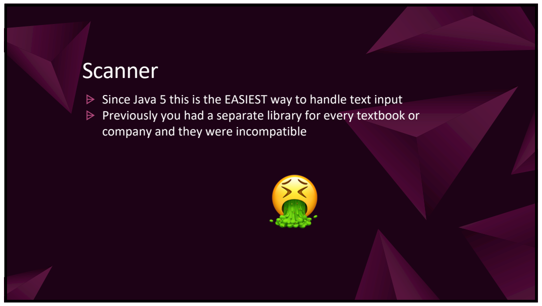
Java Scanner

CS 1400 2024
Cody Henrichsen

1

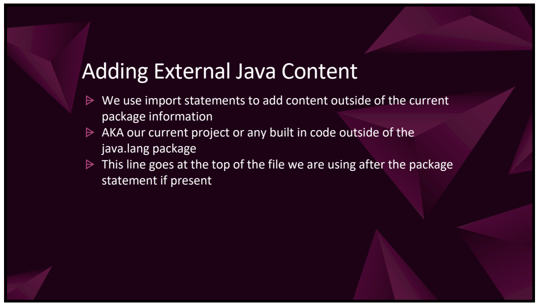


Scanner

- Since Java 5 this is the EASIEST way to handle text input
- Previously you had a separate library for every textbook or company and they were incompatible

🤮

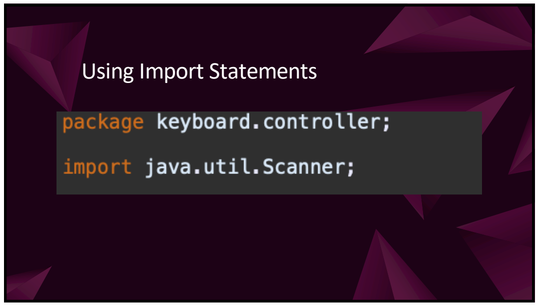
2



Adding External Java Content

- We use import statements to add content outside of the current package information
- AKA our current project or any built in code outside of the java.lang package
- This line goes at the top of the file we are using after the package statement if present

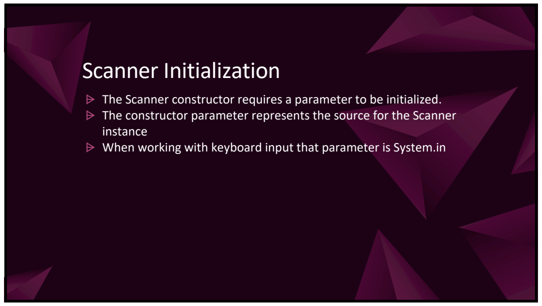
3



Using Import Statements

```
package keyboard.controller;

import java.util.Scanner;
```

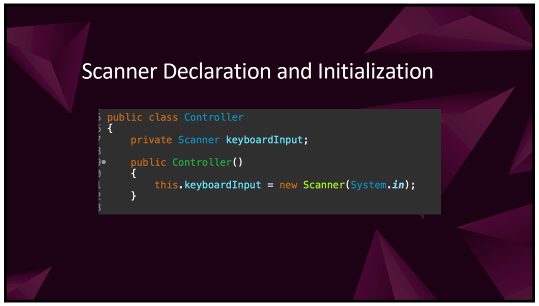
4



Scanner Initialization

- The Scanner constructor requires a parameter to be initialized.
- The constructor parameter represents the source for the Scanner instance
- When working with keyboard input that parameter is System.in

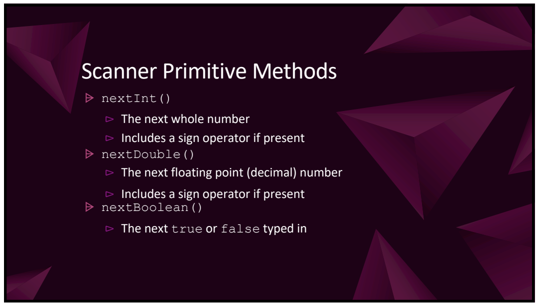
5



Scanner Declaration and Initialization

```
public class Controller
{
    private Scanner keyboardInput;

    public Controller()
    {
        this.keyboardInput = new Scanner(System.in);
    }

```

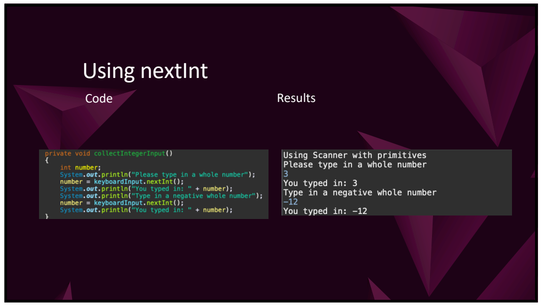
6



Scanner Primitive Methods

- `nextInt()`
  - The next whole number
  - Includes a sign operator if present
- `nextDouble()`
  - The next floating point (decimal) number
  - Includes a sign operator if present
- `nextBoolean()`
  - The next `true` or `false` typed in

7



Using nextInt

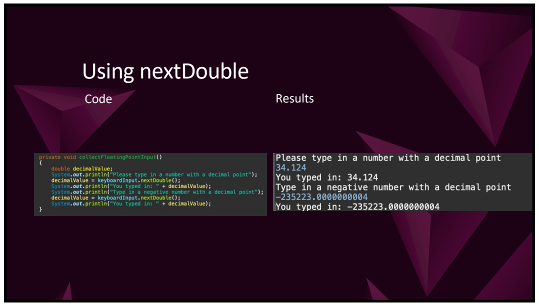
Code

```java
private void collectIntegerInput()
{
    int number;
    System.out.println("Please type in a whole number");
    number = keyboardInput.nextInt();
    System.out.println("You typed in: " + number);
    System.out.println("Type in a negative whole number");
    number = keyboardInput.nextInt();
    System.out.println("You typed in: " + number);
}
```

Results

```
Using Scanner with primitives
Please type in a whole number
3
You typed in: 3
Type in a negative whole number
-12
You typed in: -12
```

8



Using nextDouble

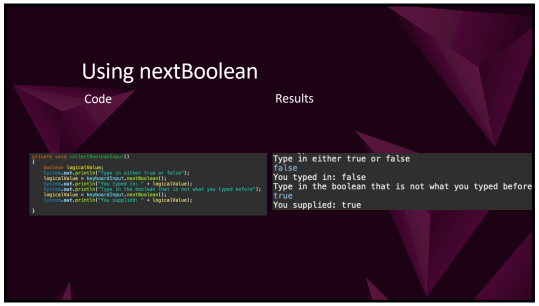
Code

```java
private void collectFloatingPointInput()
{
    double decimalValue;
    System.out.println("Please type in a number with a decimal point");
    decimalValue = keyboardInput.nextDouble();
    System.out.println("You typed in: " + decimalValue);
    System.out.println("Type in a negative number with a decimal point");
    decimalValue = keyboardInput.nextDouble();
    System.out.println("You typed in: " + decimalValue);
}
```

Results

```
Please type in a number with a decimal point
34.124
You typed in: 34.124
Type in a negative number with a decimal point
-235223.0000000004
You typed in: -235223.0000000004
```

9



Using nextBoolean

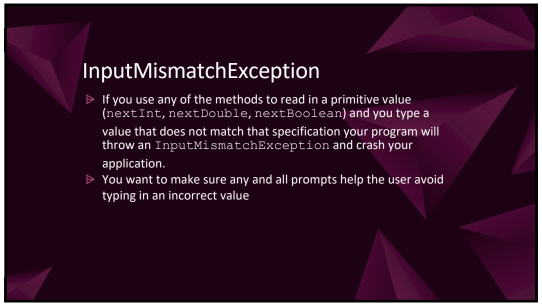
Code

```
private void collectBooleanInput()
{
    boolean logicalValue;
    System.out.println("Type in either true or false");
    logicalValue = keyboardInput.nextBoolean();
    System.out.println("You typed in: " + logicalValue);
    System.out.println("Type in the boolean that is not what you typed before");
    logicalValue = keyboardInput.nextBoolean();
    System.out.println("You supplied: " + logicalValue);

}
```

Results

```
Type in either true or false
false
You typed in: false
Type in the boolean that is not what you typed before
true
You supplied: true
```

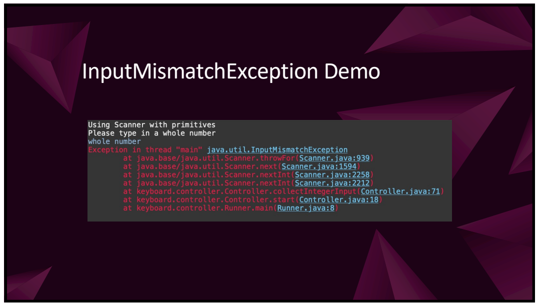
10



InputMismatchException

- If you use any of the methods to read in a primitive value (`nextInt`, `nextDouble`, `nextBoolean`) and you type a value that does not match that specification your program will throw an `InputMismatchException` and crash your application.
- You want to make sure any and all prompts help the user avoid typing in an incorrect value

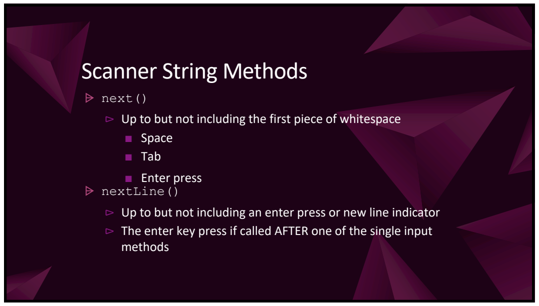
11



InputMismatchException Demo

```
Using Scanner with primitives
Please type in a whole number
whole number
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 keyboard.controller.Controller.collectIntegerInput(Controller.java:71)
	at keyboard.controller.Controller.start(Controller.java:18)
	at keyboard.controller.Runner.main(Runner.java:8)
```

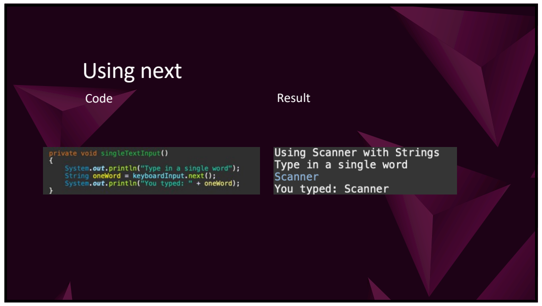
12



Scanner String Methods

- `next()`
  - Up to but not including the first piece of whitespace
    - Space
    - Tab
    - Enter press
- `nextLine()`
  - Up to but not including an enter press or new line indicator
  - The enter key press if called AFTER one of the single input methods

13



Using next

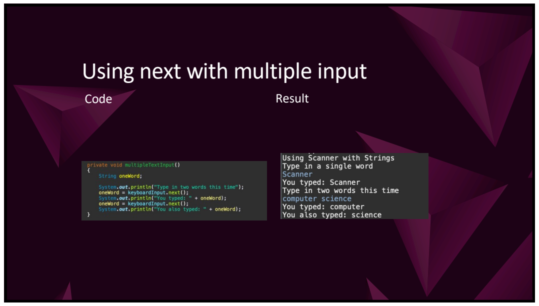
Code

```java
private void singleTextInput()
{
    System.out.println("Type in a single word");
    String oneWord = keyboardInput.next();
    System.out.println("You typed: " + oneWord);
}
```

Result

```
Using Scanner with Strings
Type in a single word
Scanner
You typed: Scanner
```

14



Using next with multiple input

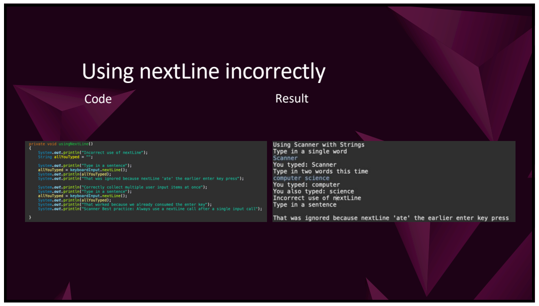
Code

```java
private void multipleTextInput()
{
    String oneWord;

    System.out.println("Type in two words this time");
    oneWord = keyboardInput.next();
    System.out.println("You typed: " + oneWord);
    oneWord = keyboardInput.next();
    System.out.println("You also typed: " + oneWord);
}
```

Result

```
Using Scanner with Strings
Type in a single word
Scanner
You typed: Scanner
Type in two words this time
computer science
You typed: computer
You also typed: science
```

15



Using nextLine incorrectly

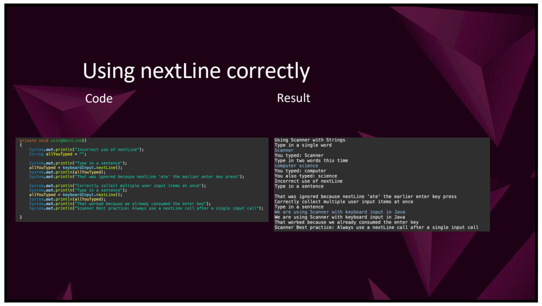
Code

```java
private void usingNextLine()
{
    System.out.println("Incorrect use of nextLine");
    String allYouTyped = "";

    System.out.println("Type in a sentence");
    allYouTyped = keyboardInput.nextLine();
    System.out.println(allYouTyped);
    System.out.println("That was ignored because nextLine 'ate' the earlier enter key press");

    System.out.println("Correctly collect multiple user input items at once");
    System.out.println("Type in a sentence");
    allYouTyped = keyboardInput.nextLine();
    System.out.println(allYouTyped);
    System.out.println("That worked because we already consumed the enter key");
    System.out.println("Scanner Best practice: Always use a nextLine call after a single input call");

}
```

Result

```
Using Scanner with Strings
Type in a single word
Scanner
You typed: Scanner
Type in two words this time
computer science
You typed: computer
You also typed: science
Incorrect use of nextLine
Type in a sentence

That was ignored because nextLine 'ate' the earlier enter key press
```

16



Using nextLine correctly

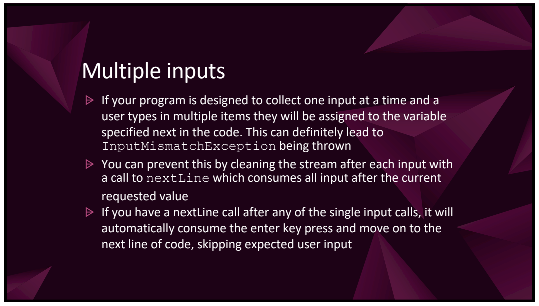
Code

```java
private void usingNextLine()
{
    System.out.println("Incorrect use of nextLine");
    String allYouTyped = "";

    System.out.println("Type in a sentence");
    allYouTyped = keyboardInput.nextLine();
    System.out.println(allYouTyped);
    System.out.println("That was ignored because nextLine 'ate' the earlier enter key press");

    System.out.println("Correctly collect multiple user input items at once");
    System.out.println("Type in a sentence");
    allYouTyped = keyboardInput.nextLine();
    System.out.println(allYouTyped);
    System.out.println("That worked because we already consumed the enter key");
    System.out.println("Scanner Best practice: Always use a nextLine call after a single input call");

}
```

Result

```
Using Scanner with Strings
Type in a single word
Scanner
You typed: Scanner
Type in two words this time
computer science
You typed: computer
You also typed: science
Incorrect use of nextLine
Type in a sentence

That was ignored because nextLine 'ate' the earlier enter key press
Correctly collect multiple user input items at once
Type in a sentence
We are using Scanner with keyboard input in Java
We are using Scanner with keyboard input in Java
That worked because we already consumed the enter key
Scanner Best practice: Always use a nextLine call after a single input call
```

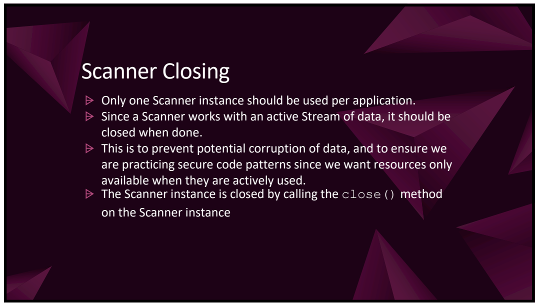
17



Multiple inputs

- If your program is designed to collect one input at a time and a user types in multiple items they will be assigned to the variable specified next in the code. This can definitely lead to `InputMismatchException` being thrown
- You can prevent this by cleaning the stream after each input with a call to `nextLine` which consumes all input after the current requested value
- If you have a nextLine call after any of the single input calls, it will automatically consume the enter key press and move on to the next line of code, skipping expected user input

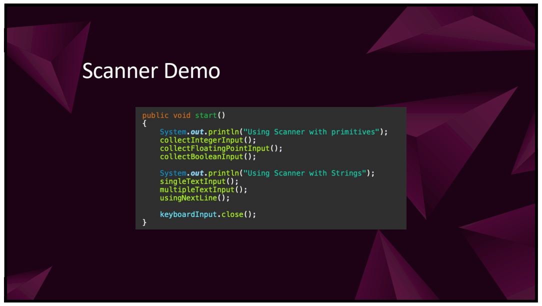
18



Scanner Closing

- Only one Scanner instance should be used per application.
- Since a Scanner works with an active Stream of data, it should be closed when done.
- This is to prevent potential corruption of data, and to ensure we are practicing secure code patterns since we want resources only available when they are actively used.
- The Scanner instance is closed by calling the `close()` method on the Scanner instance

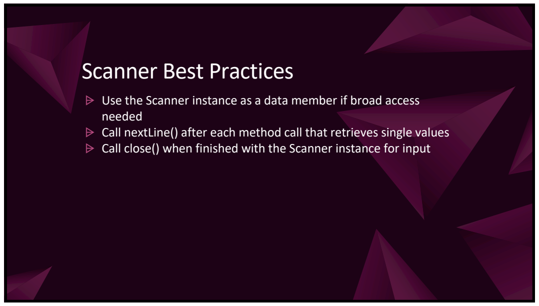
19



Scanner Demo

```
public void start()
{
    System.out.println("Using Scanner with primitives");
    collectIntegerInput();
    collectFloatingPointInput();
    collectBooleanInput();

    System.out.println("Using Scanner with Strings");
    singleTextInput();
    multipleTextInput();
    usingNextLine();

    keyboardInput.close();
}
```

20



Scanner Best Practices

- Use the Scanner instance as a data member if broad access needed
- Call nextLine() after each method call that retrieves single values
- Call close() when finished with the Scanner instance for input

21