Oracle Academy
Java for AP Computer Science A

2-2

What is my Program Doing?

ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Understand how Java is read line by line
  - Set and use breakpoints
  - End statements with semicolons (;)
  - Organize code using whitespace and other conventions
  - Create comments

Reading a Program Line by Line

- Each line in a program is read one at a time

```
1  System.out.println("Line 1");
2  System.out.println("Line 2");
3  System.out.println("Line 3");
4  System.out.println("Line 4");
5  System.out.println("Line 5");
```

- In the example…
  - Line 1 is read…
  - Then Line 2…
  - Then Line 3…
  - Then Line 4…
  - Then Line 5…

Reading Line by Line

- Java is mostly read line by line

- But there are a few additional points to consider

- We'll investigate using...
  - A breakpoint
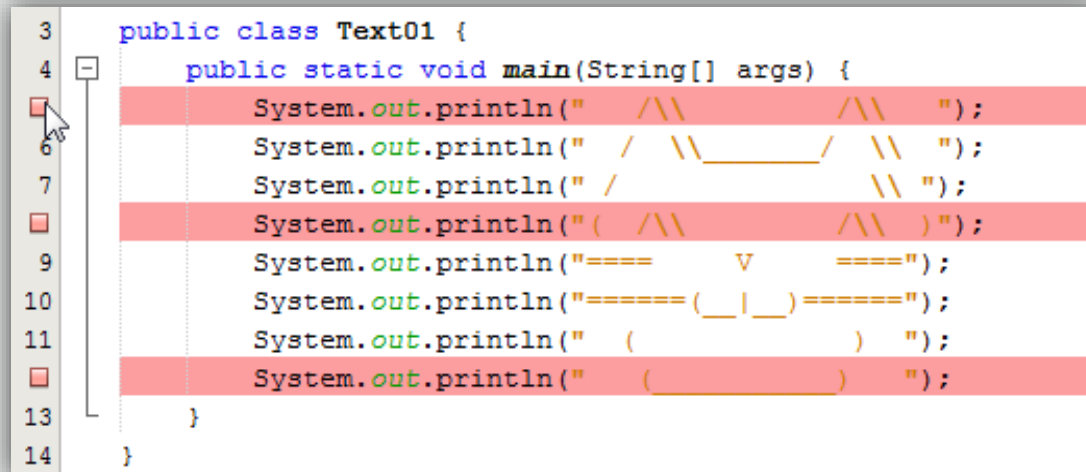 - Other features of a Java IDE

Breakpoints

- Set a breakpoint in your code to
  - Pause code execution
  - Check the current state of the program
  - Help debug
- Breakpoints affect code execution …
  - When code is run with the debugger

- Breakpoints can't affect code execution …
  - When code is run normally

Setting a Breakpoint Animation

- To set a breakpoint …
  - Place your cursor over a number in the left margin
  - Click … and you have a breakpoint!
 - Click again to remove a breakpoint
  - You can set many breakpoints


```java
3   public class Text01 {
4       public static void main(String[] args) {
        System.out.println("    /\\          /\\   ");
6       System.out.println("   /  \_____/  \\  ");
7       System.out.println("  /               \\ ");
        System.out.println("(    /\\          /\\   )");
9       System.out.println("====      V      ====");
10      System.out.println("======(__|__)======");
11      System.out.println("   (              )   ");
        System.out.println("     (_____)     ");
13      }
14  }
```

ORACLE
Academy

Exercise 1, Part 1

- Create a new project and add the `Text01.java` file to the project

- Set a breakpoint at Line 5 (the line with the first print statement)

- Run the program normally
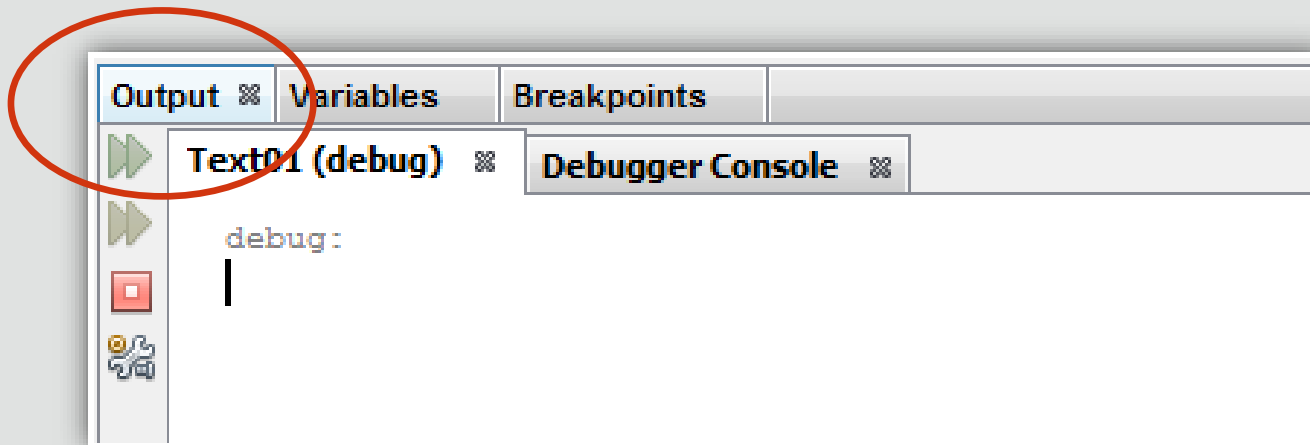 - Breakpoints should have no affect
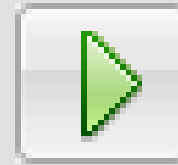
Exercise 1, Part 2

- Run the program with the debugger:
  - Make sure the Output window is showing
  - Press Step Over to go to each next line
- Observe the cat appear one line at a time

Step Over

ORACLE Academy

Exercise 1, Part 3

- Modify the code so that the first three print statements all appear on Line 5 (This is called removing whitespace)

- Run the program with the debugger:
  - Make sure the Output window is showing
  - Press Step Over Expression to go to each next line
  - Ignore the complicated code at the end of debugging
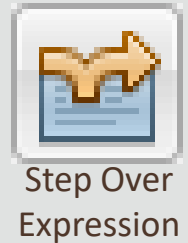
Step Over Expression

- Observe the cat appear one line at a time

- Try removing a semicolon while debugging the program

What if you do it **before** debugging?

Investigation Results, Part 1

- You could say Java reads code line by line ...
- But if multiple statements are on a single line, it's more accurate to say Java reads statement by statement
- A semicolon (;) is required to end a statement
  - Forgetting a semicolon is a common mistake
  - Other languages (Python) may not care about semicolons

```
System.out.println("Meow") ;
```

- Editing code has no affect while the program is running
- You must recompile for changes to take affect

Investigation Results, Part 2

- Java isn't precise about whitespace

- Other languages (Python) may be extremely precise

- You could write an entire program in a single line
  - But this is messy and almost impossible to work with
  - Use whitespace to keep code organized

```
3   public class Text01 {
4       public static void main(String[] args) {
5           System.out.println("   /\\          /\\    ");System.out.println("  /  \_____/  \\  ");
6   System.out.println(" /              \\ ");
7           System.out.println("(   /\\          /\\  )");
8       System.out.println("====    V    ====");
9           System.out.println("======(__|__)======");
10  System.   out.  println("  (          )  ");
11
12          System.out.println("   (_____)   ");
13      }
14  }
```

This code works... but it's super messy

Whitespace

- Whitespace is any spacing without code:
  - Space between words
  - Blank lines
  - Indentation before a line of code

```
3   public class Text01 {
4       public static void main(String[] args) {
5           System.out.println("    /\\          /\\    ");
6
7
8           System.out.println("   /  \_____/  \\   ");
9           System.out.println(" /              \\ ");
10      }
11  }
```

Effects of Whitespace

- Whitespace helps keep code organized
- Whitespace doesn't affect how code runs
- You can use whitespace however you prefer
- But proper indentation is strongly encouraged because it …
  - Prevents readability difficulties
  - Prevents mistakes while programming

Aah! Messy code!

Indentation and Curly Braces

- Indent by an additional tab (4 spaces) following an opening curly brace (**{**)

- Stop indenting by an additional tab (4 spaces) prior to a closing curly brace (**}**)

- Code within curly braces is called a block of code
  - When you add an opening curly brace (**{**) …
  - You'll eventually need a closing curly brace (**}**)
  - Mismatching or forgetting a curly brace is a common mistake

Block Example

```
public class Example
{
    public static void main(String[] args){
        System.out.println("Inner code");
        System.out.println("Inner code");
        {
            System.out.println("Inner-inner code");
        }
    }
}
```

These curly braces also create a block within a block ...

Whose code is indented further.

IDE Indentation Assistance

- An IDE may…
  - Color-code the scope of a block (Greenfoot, BlueJ)
  - Automatically indent following a curly brace
  - Highlight a matching curly brace (shown below)
- Some Java commands require curly braces, although you can always add more

```java
public class Example
{
    public static void main(String[] args){
        System.out.println("Inner code");
        System.out.println("Inner code");
        {
            System.out.println("Inner-inner code");
        }
    }
}
```

ORACLE
Academy

Exercise 2

- Create a new project and add the `Text02.java` file to the project

- Can you fix this program and produce the following output?

```
1
2
3
4
```

- Hints:
  - Your IDE underlines problematic code
  - Your IDE can highlight matching curly braces
  - Your IDE has a shortcut to correct indentation

Comments

- Neatly spaced programs can grow large and become difficult to read

- You can add comments to code to …
  - Provide an explanation or additional information to the programmer (Commenting code)
  - Disable code and prevent it from executing without erasing it (Commenting out code)

Aah! What is all this code doing?

Adding Comments to Code

- Single-line comments …
  - Start with two slashes //
  - End when the line ends

```
//A single line comment automatically ends when the line ends
System.out.println("This line prints");
```

Adding Comments to Code

- Multi-line (block)comments ...
  - Start with a slash-star   /*
  - End with a star-slash   */

```java
/*A multi line comment...
continues for many lines...
System.out.println("This line does not print");
until the star-slash appears*/
System.out.println("This line prints");
```

ORACLE
Academy

Adding Comments to Code

- Javadoc Comments
  - Start with a slash-star-star   /**
  - End with a star-slash   */
- Used to create API documentation for classes in Java programs

```
    /** A Javadoc comment is used to create class documentation
        For example, information pertaining to class methods and
        behaviors.
 In addition, usage of @ is included to document parameters.
 */
```

- For more information, see

https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

ORACLE
Academy

Reading Line by Line

- We can do a little more investigating
- We'll investigate using …
  - Code blocks
  - Comments
  - Breakpoints
  - Other features of your IDE

Exercise 3

- Create a new project and add the `Text03.java` file to the project

- Set a breakpoint at Line 11

- Run the program with the debugger:
  - Be sure to have the Output window selected
  - Press Step Over to go to each next line

Step Over

- Observe the cat face appear, but the legs don't appear

- Type drawLegs(); in Line 19 and debug the program
  - Where could you add a breakpoint to see the legs drawn one line at a time?
  - What happens to the output when lines are commented out?

Investigation Results, Part 3

- When Java reads line by line …
- It starts within the special block of code known as the main method

```java
public static void main(String[] args){

}//end method main
```

- No other code executes unless it's called
  - In this exercise, the main method must specifically call the block of code that prints legs
- Commented code is ignored
  - Comments are removed in bytecode

The Program's Flow

1. All Java programs start in the main method
2. No other code executes unless it's called

2) Then go here

1) Start here

```
public class Text03 {
    public static void drawLegs(){
        System.out.println("      ||        ||        ");
        System.out.println("      ||        ||        ");
        System.out.println("     (||)     (||)       ");
    }

    public static void main(String[] args) {
        System.out.println("    /\\            /\\   ");
        System.out.println("   /  \_____/  \\  ");
        System.out.println("  /                \\ ");
        System.out.println("(    /\\          /\\   )");
        System.out.println("====      V       ====");
        System.out.println("======(__|__)=====");
        System.out.println("    (              )   ");
        System.out.println("     (_____)     ");
        drawLegs();
    }
}
```

The Main Method

- The main method is a special block of code

- All Java programs start in the main method

- Your programs should have only 1 main method

- Methods are discussed more in the next lesson
  - drawLegs() is an example of a method

```java
public static void main(String[] args){
    //Your program starts here
}//end method main
```

Summary

- Common mistakes:
  - Missing semicolon (;)

```
System.out.println("Meow")
```

  - Mismatched {curly braces}

```
{
    System.out.println("Meow");
```

- Keep code organized using:
  - Whitespace
  - Curly Braces (**{ }**)
  - Comments

ORACLE
Academy

Summary

- In this lesson, you should have learned how to:
  - Understand how Java is read line by line
  - Set and use breakpoints
  - End statements with semicolons (;)
  - Organize code using whitespace and other conventions
  - Create comments

ORACLE
Academy