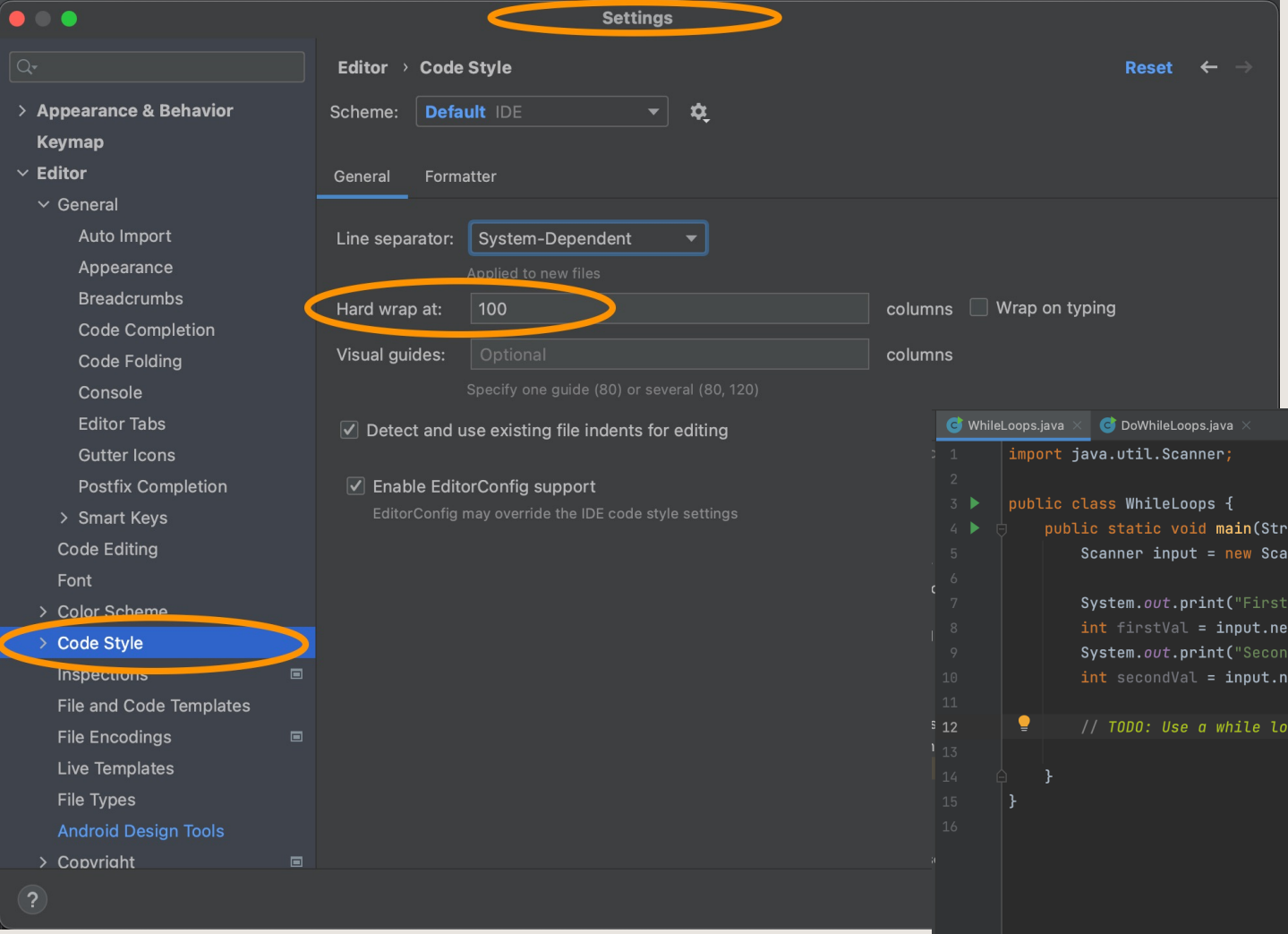
CS1101 Programming and Problem Solving

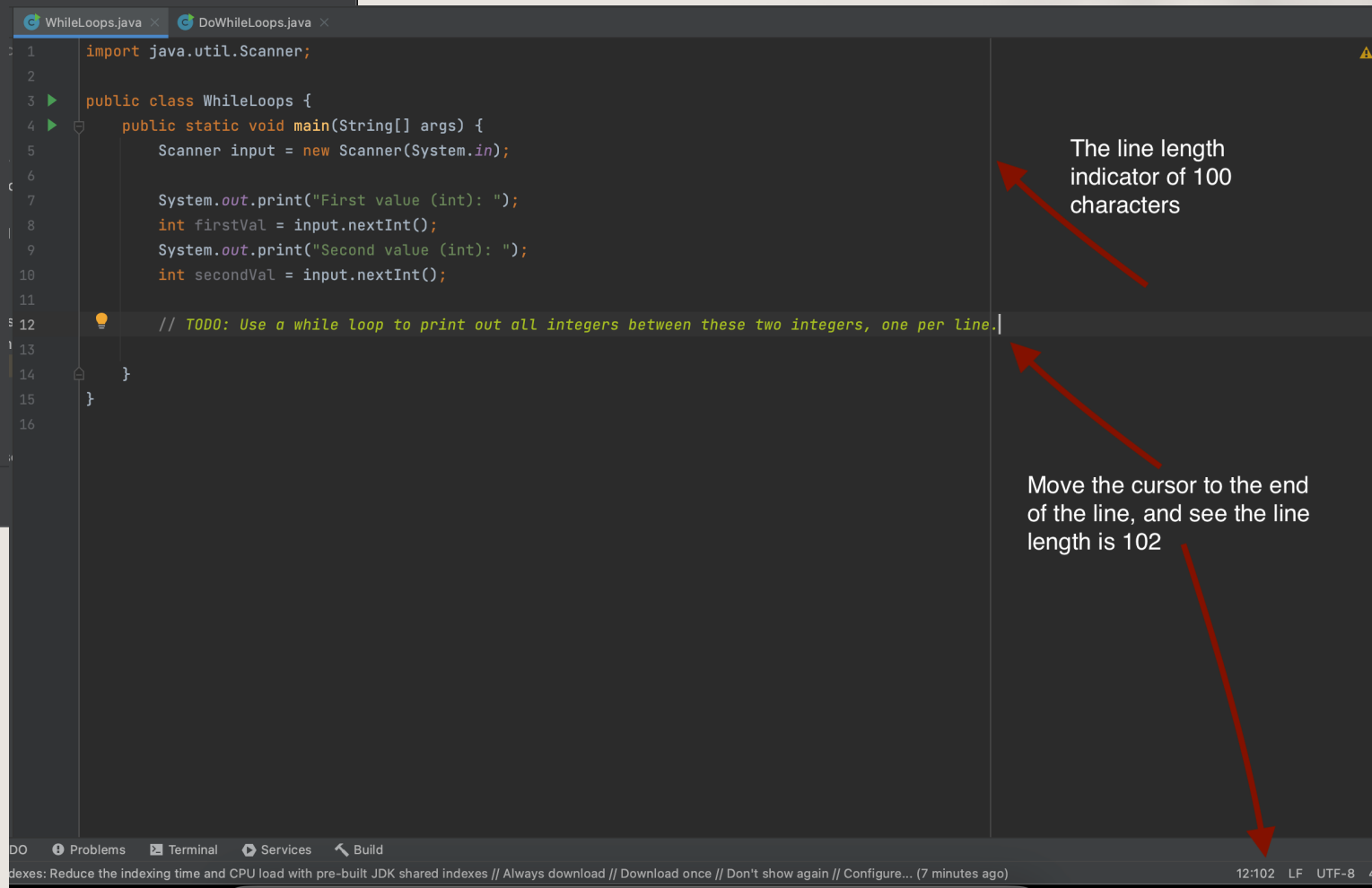
Dr. Gina Bai

Spring 2023

Logistics

- **ZY-4B** on **zyBook > Assignments**
  - Due: **Wednesday, Feb 22**, at 11:59pm
- **PA05 – W, A, B** on **zyBook > Chap 11**
  - Due: **Thursday, Feb 23**, at 11:59pm
- Midterm Exam 1 regrade requests due Tuesday, Feb 28


Settings
Editor › Code Style
Reset
Scheme: Default IDE
General
Formatter
Line separator: System-Dependent
Applied to new files
Hard wrap at: 100 columns
Wrap on typing
Visual guides: Optional columns
Specify one guide (80) or several (80, 120)
Detect and use existing file indents for editing
Enable EditorConfig support
EditorConfig may override the IDE code style settings
Appearance & Behavior
Keymap
Editor
General
Auto Import
Appearance
Breadcrumbs
Code Completion
Code Folding
Console
Editor Tabs
Gutter Icons
Postfix Completion
Smart Keys
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Android Design Tools
Copyright


WhileLoops.java
DoWhileLoops.java
import java.util.Scanner;
public class WhileLoops {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
System.out.print("First value (int): ");
int firstVal = input.nextInt();
System.out.print("Second value (int): ");
int secondVal = input.nextInt();
// TODO: Use a while loop to print out all integers between these two integers, one per line.
}
}
The line length indicator of 100 characters
Move the cursor to the end of the line, and see the line length is 102
Problems
Terminal
Services
Build
12:102
LF
UTF-8

for Loop

zyBook Chap 5.4, 5.5, 5.6

Indefinite vs. Definite Loops

- **Indefinite loop**
  - A loop that keeps looping **as long as a condition is true**, or until a condition becomes false
    - while loops, do-while loops

- **Definite loop**
  - A loop that executes a **known number** of times
    - for loops

while Loop

```
<initialization>;
while (<continuation test>) {
    <controlled statement(s)>;
    <update>;
}
<statement(s)>;
```

```
int i = 0;
while (i < 3){
    System.out.print(i + " ");
    ++i;
}
```

```
for (int i = 0; i < 3; ++i){
    System.out.print(i + " ");
}
```

```
int i;
for (i = 0; i < 3; ++i){
    System.out.print(i + " ");
}
```

```
0 1 2
```

for Loop

```
for (<initialization>; <continuation test>; <update>) {
    <controlled statement(s)>;
}
<statement(s)>;
```

Step 1: Initialize the loop counter

- Usually, **declare + initialize**
- **Performed once** as the loop begins
- Can start at any value
- Can use any name (e.g., i, j, k…)

Step 2: Check the continuation test

- Tests the loop counter variable against a limit
- Uses relational operators (<, >, <=, >=)

Step 3: If true, execute the controlled stmt(s)

Step 4: Update the loop counter

- E.g., +=, -=, *=, /=, ++, --

Looping: Check the continuation test again…

```
for (int i = 0; i < 3; ++i){
    System.out.print(i + " ");
}
```

i = 0;

i < 3 ?
True
Print out i // 0
++i // i = 1;

i < 3 ?
True
Print out i // 1
++i // i = 2;

i < 3 ?
True
Print out i // 2
++i // i = 3;

i < 3 ?
False

Q: What's the output of the following code?

```
int n = 5;
for (int i = 1; i <= n; ++i) {
    System.out.print(i + " ");
}
```

Ouput: 1 2 3 4 5

```
int n = 5;
for (int i = 0; i < n; ++i) {
    System.out.print(i + " ");
}
```

Ouput: 0 1 2 3 4

```
int n = 5;
for (int i = n; i >= 1; --i) {
    System.out.print(i + " ");
}
```

Ouput: 5 4 3 2 1

TopHat: What's the output of the following code?

```
int x = 3;
for (int i = 1; i <= 10; ++i) {
    System.out.println(x);
}
System.out.println(i);
```

Compile Error
Since the loop counter `i` is **declared** and **initialized** in the loop header, the scope of `i` is within the for loop

Loops and Strings

```java
import java.util.Scanner;

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

        Scanner console  = new Scanner(System.in);
        System.out.print("Enter String: ");
        String inputString = console.nextLine();

        // Loop through string, and
        // print out one index and character on each line
        // Reminder: String index starts at 0
        for (int i = 0; ____________________________; i++) {
            System.out.println(i + ": " + inputString.charAt(i));
        }
    }
}
```

```
$ javac StringLoop.java
$ java StringLoop
Enter String: CS 1101
0: C
1: S
2:
3: 1
4: 1
5: 0
6: 1
```

Loops and Strings

```java
import java.util.Scanner;

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

        Scanner console  = new Scanner(System.in);
        System.out.print("Enter String: ");
        String inputString = console.nextLine();

        // Loop through string, and
        // print out one index and character on each line
        // Reminder: String index starts at 0
        for (int i = 0; i < inputString.length(); i++) {
            System.out.println(i + ": " + inputString.charAt(i));
        }
    }
}
```

```
$ javac StringLoop.java
$ java StringLoop
Enter String: CS 1101
0: C
1: S
2:
3: 1
4: 1
5: 0
6: 1
```

Nested Loop

zyBook Chap 5.7

Nested Loops – A loop inside of another loop

```
while (<condition>) {
    <statements>;
    for (<initialization>; <continuation test>; <update>) {
        <statements>;
    }
}
<statement>;
```

```
for (<initialization>; <continuation test>; <update>) {
    <statements>;
    for (<initialization>; <continuation test>; <update>) {
        <controlled statements>;
    }
}
<statement>;
```

Example – 1

"Sentinel Loop"
Use a **while** loop since we do not know when the sentinel value would be entered by the user.

```
import java.util.Scanner;

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

        Scanner console  = new Scanner(System.in);
        System.out.print("Enter String (or QUIT to quit): ");
        String inputString = console.nextLine();

        while(!inputString.equals("QUIT")) {
            for (int i = 0; i < inputString.length(); i++) {
                System.out.println(i + ": " + inputString.charAt(i));
            }
            System.out.print("Enter String (or QUIT to quit): ");
            inputString = console.nextLine();
        }
    }
}
```

```
$ javac StringLoop.java
$ java StringLoop
Enter String (or QUIT to quit): Hello
0: H
1: e
2: l
3: l
4: o
Enter String (or QUIT to quit): World
0: W
1: o
2: r
3: l
4: d
Enter String (or QUIT to quit): Quit
0: Q
1: u
2: i
3: t
Enter String (or QUIT to quit): QUIT
```

Example – 2

```
public class NestedFor{
    public static void main(String[] args){
        int row = 4;
        int column = 10;

        // Outer for loop
        // 4 rows of the pattern
        for (int i = 0; i < row; ++i) {
            // Inner for loop
            // in each row, 10 repetitions of printing j
            for (int j = 0; j < column; ++j) {
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

```
$ javac NestedFor.java
$ java NestedFor
0123456789
0123456789
0123456789
0123456789
```

i = 0
i < 4 ?
True

j = 0
j < 10 ?
True
Print out j
++j // **j = 1**

j < 10 ?
True
Print out j
++j // **j = 2**

…

j < 10 ?
True
Print out j
++j // **j = 10**

j < 10 ?
False

Move the cursor
to a new line
++i; // **i = 1**

i < 4 ?
True

j = 0
j < 10 ?
True
Print out j
++j // **j = 1**

…

j < 10 ?
False

Move the cursor
to a new line
++i; // **i = 2**

…
…

Coding Practice

Q: Reproduce the following patterns with nested for loops

```
*
**
***
****
*****
```

```
    1
   2
  3
 4
5
```

Hint: print out the spaces and then the number

Sample Solutions

```
*
**
***
****
*****
```

```
    1
   2
  3
 4
5
```

```
int row = 5;

for (int i = 0; i < row; ++i) {
    for (int j = 0; j <= i; ++j) {
        System.out.print("*");
    }
    System.out.println();
}
```

```
int row = 5;
int column = 5;

for (int i = 1; i <= row; ++i) {
   for (int j = 0; j < column - i; ++j) {
       System.out.print(" ");
   }
   System.out.println(i);
}
```