CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **PA03 – W, A, B** on **zyBook > Chap 11**
  - Due: **Saturday, Feb 11**, at 11:59pm
- **ZY-4A** on **zyBook > Assignments**
  - Due: **Wednesday, Feb 15**, at 11:59pm
- **PA04 – W1, W2, A, B** on **zyBook > Chap 11**
  - Due: **Thursday, Feb 16**, at 11:59pm

FAQ – next() vs. nextLine()

- What we see:

```
A  B
C
```

- What Scanners see:

```
A  B\nC
```

next() next() nextLine()

next()
grabs one token a time, e.g., **A**
***Note: Tokens are separated by whitespaces, such as a space, a tab, a newline

nextLine()
grabs all tokens till and including \n, e.g., **A B\n**
***Note: \n itself could be read in as a String by nextLine()

Recap – Conditionals

```
if (<condition>) {
    <controlled stmt(s)>;
}
<statement(s)>;
```

Zero or One of these sets of controlled stmts will be executed

```
if (<cond1>) {
    <cond1 controlled stmt(s)>;
} else if (<cond2>) {
    <cond2 controlled stmt(s)>;
}
<statement(s)>;
```

```
if (<condition>) {
    <if controlled stmt(s)>;
} else {
    <else controlled stmt(s)>;
}
<statement(s)>;
```

Exactly One of these sets of controlled stmts will be executed

```
if (<cond1>) {
    <cond1 controlled stmt(s)>;
} else if (<cond2>) {
    <cond2 controlled stmt(s)>;
} else {
    <else controlled stmt(s)>;
}
<statement(s)>;
```

The Use of Curly Braces

- The curly braces are **required** if there are **multiple statements** on a branch
- The curly braces are **optional** if there is one **single statement** on a branch
- It is **highly recommended** to **always** use braces, even when not necessary

```
if (<condition>){
    <if controlled statement(s)>;
} else {
    <else controlled statement(s)>;
}
```

Valid, and recommended

```
if (<condition>)
    <if controlled SINGLE statement>;
else
    <else controlled SINGLE statement>;
```

Valid, but NOT recommended

Example – Valid

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

        int a = 1;

        if(a > 0){
            System.out.println("1: a > 0");
        } else {
            System.out.println("1: a <= 0");
        }

        if(a > 0)
            System.out.println("2: a > 0");
        else
            System.out.println("2: a <= 0");
    }
}
```

```
$ javac SingleLineIf.java
$ java SingleLineIf
1: a > 0
2: a > 0
```

Example – Invalid

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

        int a = 1;

        if(a > 0) {
            System.out.println("a > 0");}
            System.out.println("a is " + a);
        else
            System.out.println("a <= 0");
    }
}
```

```
$ javac SingleLineIf.java
SingleLineIf.java:9: error: 'else' without 'if'
        else
        ^
1 error
```

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

        int a = 1;

        if(a > 0)
            System.out.println("a > 0");
        else {
            System.out.println("a <= 0");}
            System.out.println("a is " + a);
    }
}
```

```
$ javac SingleLineIf.java
$ java SingleLineIf
a > 0
a is 1
```

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

```
int a = 1, b = 2, c = 3, d = 4;
int e = 5, f = 6, g = 7, h = 8;

if (a > b)
    if (c > d)
        e = f;
else
   g = h;

System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
System.out.println(e);
System.out.println(f);
System.out.println(g);
System.out.println(h);
```

```
1
2
3
4
5
6
7
8
```

Q: Why the value of g is not updated?

“Dangling else” Problem

- Every `else`-part is paired with the **nearest unmatched** `if`-part
- Computers ignore the indentation

Original code:

```
if (a > b)
    if (c > d)
        e = f;
else
   g = h;
```

Computers read the code as:

```
if (a > b) {
    if (c > d) {
        e = f;
    } else {
        g = h;
    }
}
```

Returning Within a Conditional

Returning Within a Conditional

- When a **return statement** is reached, the specified **value is returned**, and we **exit the method**
  - Any remaining portion of the method is NOT executed

- **Must return on ALL paths out of a method**

Recap – Conditionals

```
if (<condition>) {
    <controlled stmt(s)>;
}
<statement(s)>;
```

Zero or One of these sets of controlled stmts will be executed

```
if (<cond1>) {
    <cond1 controlled stmt(s)>;
} else if (<cond2>) {
    <cond2 controlled stmt(s)>;
}
<statement(s)>;
```

```
if (<condition>) {
    <if controlled stmt(s)>;
} else {
    <else controlled stmt(s)>;
}
<statement(s)>;
```

Exactly One of these sets of controlled stmts will be executed

```
if (<cond1>) {
    <cond1 controlled stmt(s)>;
} else if (<cond2>) {
    <cond2 controlled stmt(s)>;
} else {
    <else controlled stmt(s)>;
}
<statement(s)>;
```

Q: Will the following code compile and run?

```java
public class ReturnConditional {
    public static void main(String[] args) {
        System.out.println(isPositive(-2));
    }

    public static String isPositive(int a) {
        if (a > 0 ) {
            return "Positive";
        } else if ( a < 0 ) {
            return "Negative";
        } else if ( a == 0 ) {
            return "Zero";
        }
    }
}
```

Zero or One of these sets of controlled statements will be executed

Given the conditional structure, the compiler believes it is possible that none of the return statements will be reached.

```
$ javac ReturnConditional.java
ReturnConditional.java:15: error: missing return statement
    }
    ^
1 error
```

Corrected implementation:

```java
public class ReturnConditional {
    public static void main(String[] args) {
        System.out.println(isPositive(-2));
    }

    public static String isPositive(int a) {
        if (a > 0 ) {
            return "Positive";
        } else if ( a < 0 ) {
            return "Negative";
        } else {
            return "Zero";
        }
    }
}
```

```
$ javac ReturnConditional.java
$ java ReturnConditional
Negative
```

More examples (equivalent implementations)

```java
/**
 * Returns the max of x and y
 * @param  x integer to compare
 * @param  y integer to compare
 * @return  the max of x and y
 */
public static int maxA(int x, int y) {
    int max = y;
    if (x > y) {
        max = x;
    }
    return max;
}
```

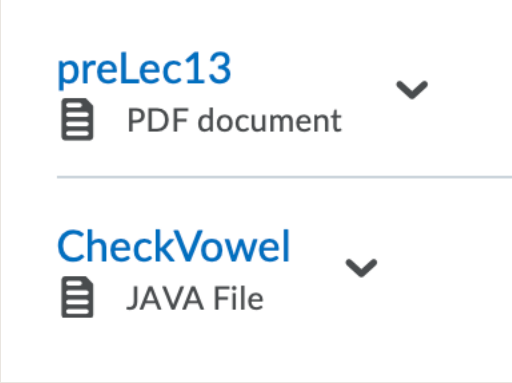
```java
public static int maxB(int x, int y) {
    if (x > y) {
        return x;
    } else {
        return y;
    }
}
```

```java
public static int maxC(int x, int y) {
    if (x > y) {
        return x;
    }
    return y;
}
```

Coding Practice

Complete the program **CheckVowel** that

- prompts the user for an input String
- determines if the input String
  - starts with a vowel
  - ends with a vowel
  - starts and ends with vowels



Sample Solution

```java
import java.util.Scanner;

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

        Scanner input = new Scanner(System.in);
        System.out.print("Enter a String: ");
        String str = input.next();

        // String index starts at 0
        char first = str.charAt(0);
        char last = str.charAt(str.length() - 1);

        if( isVowel(first) && isVowel(last) ) {
            System.out.print("The input " + str + " starts and ends with vowels.");
        } else if( isVowel(last) ) {
            System.out.print("The input " + str + " ends with a vowel.");
        } else if( isVowel(first) ) {
            System.out.print("The input " + str + " starts with a vowel.");
        } else {
            System.out.print("The input is " + str + ".");
        }
    }

    public static boolean isVowel(char letter) {
        // Use double equal signs to compare primitive data
        return letter == 'A' || letter == 'a' ||
               letter == 'E' || letter == 'e' ||
               letter == 'I' || letter == 'i' ||
               letter == 'O' || letter == 'o' ||
               letter == 'U' || letter == 'u';
    }
}
```