CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **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
- **ZY-4B** on **zyBook > Assignments**
  - Due: **Wednesday, Feb 22**, at 11:59pm

"Boolean Zen"
Better Programming Style

Boolean Method

```java
public static boolean isEven(int num) {

    boolean isEven;

    if (num % 2 == 0) {
        isEven = true;
    } else {
        isEven = false;
    }
    return isEven;
}
```

Correct, but verbose

```java
public static boolean isEven(int num) {

    if (num % 2 == 0) {
        return true;
    } else {
        return false;
    }
}
```

Better

```java
public static boolean isEven(int num) {
    return num % 2 == 0;
}
```

Perfect

"Boolean Zen" Template

Replace...

```
public static boolean methodName(parameter(s)) {

    if ( <expression> ) {
        return true;
    } else {
        return false;
    }
}
```

With...

```
public static boolean methodName(parameter(s)) {
    return <expression>;
}
```

Boolean Method Call

```java
if ( isEven(number) == true ) {
    System.out.println("Even");
}

if ( isEven(number) ) {
    System.out.println("Even");
}

if ( isEven(number) == false ) {
    System.out.println("Odd");
}

if ( !isEven(number) ) {
    System.out.println("Odd");
}
```

Verbose → true == true

Preferred

Verbose → false == false

Preferred

"Boolean Zen"

in CheckVowel.java (Lec13)

```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';
    }
}
```

Assertions

zyBook Chap 5.11

Assertions

- Assertion: A declarative sentence that is either true or false
  - May depend on the context
  - Examples
    - When $x = 13$, $x > 45$ (false)
    - x divided by two equals seven (depends on the value of x)
- Provable Assertion: An assertion that can be proven to be true at a particular point in program execution
  - Help simplify code
  - Understand code better

Example

```
TIP: Consider it as
System.out.print(x > 3)
e.g., at Point A, right above x--;
```

Q: What do we know about the value of x at Point A, Point B, and Point C?

```
if (x > 3) {
    // Point A: Is x > 3 Always True? Sometimes True? Never True?
    x--;
} else {
    // Point B: Is x > 3 Always True? Sometimes True? Never True?
    x++;
}
// Point C: Is x > 3 Always True? Sometimes True? Never True?
```

Always, since it's in the if part, which means the condition x >3 is met

Never, since it's in the else part, which means the condition x >3 is NOT met

Sometimes, e.g.,
If x is 3, x++ in the conditional, and becomes 4
If x is 4, x-- in the conditional, and becomes 3

Q: Identify the various assertions in the anotherSecret method as being either always true, never true, or sometimes true at various points in program execution.

```
public static int anotherSecret(int a, int b, int c) {
    int temp = c;
    if( a < b && b < c ){
        temp = b;
        b--;
        // POINT A
    } else if ( a != c ){
        // POINT B
        temp = a;
        a = c;
    }
    return temp;
}
```

	ALWAYS	NEVER	SOMETIMES
POINT A: **a < b**			✓
POINT A: **a != c**	✓		
POINT B: **a < b**			✓
POINT B: **a != c**	✓		

More String Methods

zyBook Chap 4.15

Compare two Strings

```
public boolean equals(Object anObject)
public boolean equalsIgnoreCase(String anotherString)
```

- For example,

```
String school = "Vandy";
school.equals("Vandy");    // true
school.equals("VAnDy");    // false
school.equalsIgnoreCase("VAnDy");  // true
```

Starts with, Ends with, Contains a substring

```
   String school = "Vandy";
public boolean startsWith(String prefix)
   school.startsWith("V");    // true
   school.startsWith("Va");   // true
   school.startsWith("VA");   // false
public boolean endsWith(String suffix)
   school.endsWith("dy");      // true
   school.endsWith("y");       // true
public boolean contains(String s)
   school.contains("n");      // true
   school.contains("N");      // false
```

TODO:
Rewrite CheckVowel.java in Lec13

Replace a char or a substring

```
public String replace(char oldChar, char newChar)
public String replace(String oldString, String newString)
```

- For example,

```
String cheer = "VANDY – ANCHOR DOWN";
cheer.replace('O', 'o');   // "VANDY – ANCHoR DoWN"
cheer.replace("AN", "an"); // "VanDY – anCHOR DOWN"
```

Character Operations from Character Class

zyBook Chap 4.20

Character Class

- Within the default package `java.lang`
  - The `Character` class wraps a value of the primitive type char in an object.
  - An object of type `Character` contains a single field whose type is char.

- Syntax:
  `<ClassName>.<methodName>(parameter(s))`
  `Character.<methodName>(parameter(s))`

`public static String toString(char ch)`

- Returns:
  - a String object representing the specified char.

- For example,

```
Character.toString('a');   // "a"
```

`public static int getNumericValue(char ch)`

- Returns:
  - **the numeric value of the character, as a nonnegative int value;**
  - -2 if the character has a numeric value but the value can not be represented as a nonnegative int value;
  - -1 if the character has no numeric value.

- For example,

```
Character.getNumericValue('6');  // 6
Character.getNumericValue('-6'); // Error: unclosed character literal
```

```
public static boolean isDigit(char ch)
public static boolean isLetter(char ch)
```

- For example,

```
Character.isDigit('X');     // false
Character.isDigit('9');     // true

Character.isLetter('X');    // true
Character.isLetter('9');    // false
```

```
public static boolean isLowerCase(char ch)
public static boolean isUpperCase(char ch)
```

- For example,

```
Character.isLowerCase('Q');    \\ false
Character.isLowerCase('n');    \\ true
Character.isLowerCase('!');    \\ false

Character.isUpperCase('Q');    \\ true
Character.isUpperCase('n');    \\ false
Character.isUpperCase('!');    \\ false
```

```
public static char toLowerCase(char ch)
public static char toUpperCase(char ch)
```

- For example,

```
Character.toLowerCase('Q');    \\ 'q'
Character.toLowerCase('n');    \\ 'n'
Character.toUpperCase('!');    \\ '!'

Character.toUpperCase('Q');    \\ 'Q'
Character.toUpperCase('n');    \\ 'N'
Character.toUpperCase('!');    \\ '!'
```