Sukkur Institute of Business Administration University
Department of Computer Science

Object Oriented Programming
BS – II (CS/SE/AI)
Spring 2025

Lab # 02: To become familiar with Control Structures and String

Instructor: Moona Solangi

Lab Report Rubrics (Add the points in each column, then add across the bottom row to find the total score)					Total Marks
S.No	**Criterion**	**0.5**	**0.25**	**0.125**	
1	Accuracy	☐ Desired output	☐ Minor mistakes	☐ Critical mistakes	
2	Timing	☐ Submitted within the given time	☐ 1 day late	☐ More than 1 day late	

Note: Submit this lab hand-out before the next lab with attached solved activities and exercises

Objectives

After performing this lab, students will be able to understand,

- Control Statements
- Strings in JAVA
- Java Math Class

Control Structures

A program is a list of instructions or blocks of instructions. Java provides control structures that can change the path of execution and control the execution of instructions.

There are *three* kinds of control structures in java**:**

i. **Conditional Branches**, which are used for choosing between two or more paths. There are three types in Java: if/else/else if, ternary operator and switch.

ii. **Loops** that are used to iterate through multiple values/objects and repeatedly run specific code blocks. The basic loop types in Java are: for, while and do while.

iii. **Branching Statements**, which are used to alter the flow of control in loops. There are two types in Java: "break" and "continue".

Decision Making	Loop statements	Jump statements
• **if statements** • **switch statement** **(After Java 5, strings can be used with in the switch statement)**	• **while** • **-do while** • **for loop**	• **break statement** • **continue statement**

Note: The above text in red is different in Java than C++ rest is same.

Conditional Branches:

If / else/ else if:

If a certain condition is true, then if block will be executed otherwise else block will be executed.

Theoretically, we can infinitely chain or nest if/else blocks but this will hurt code readability that's why it's not advised.

*Syntax***:**

```
class ExampleIfElseStatement
{
        public static void main(String args[])
        {
                int age = 15;
                if (age > 18) // if age is greater than 18
                {
```

```
                              // It will be print if block condition is true.
                              System.out.println("The age of person is : " + age + "He/she is
                              eligible for voting");
                    }
                    else{
                              // It will be print if block condition is false.
                              System.out.println("He/she is not eligible for voting");
                    }
          }
}
```

Ternary Operator:

We can use a ternary operator as a shorthand expression that works like an *if/else* statement.

*Syntax***:**

System.out.println(count > 2 ? "Count is higher than 2" : "Count is lower or equal than 2");

Switch:

Three or more if/else statements can be hard to read. As one of the possible workarounds, we can use switch, as seen above. If we have multiple cases to choose from, we can use a switch statement.

*Syntax***:**

```
int count = 3;
switch (count) {
case 0:
    System.out.println("Count is equal to 0");
    break;
case 1:
    System.out.println("Count is equal to 1");
    break;
default:
    System.out.println("Count is either negative, or higher than 1");
    break;
}
```

Switch with String variable

```
import java.util.*;
public class switchStatement {
        public static void main(String[] args) {
            String levelString="Expert";
            Scanner in = new Scanner(System.in);
            levelString = in.nextLine();

            int level=0;
```

```java
        switch(levelString){
            case "Beginner":
                    level=1;
            break;
            case "Intermediate":
                    level=2;
            break;
            case "Expert":
                    level=3;
            break;
            default:
                    level=0;
            break;
        }
        System.out.println("Your Level is: "+level);
    }
}
```

Loops:

Loops are very useful when a programmer wants to execute a statement or block of statements multiple times until a certain condition is true.

//For loop:
```java
for (int i = 1; i <= 50; i++) {
   System.out.println("Hello World!");
}
```

//while loop:
```java
int whileCounter = 1;
while (whileCounter <= 50) {
   System.out.println("Hello World!");
   whileCounter++;
}
```

// do while loop:
```java
class ExampleDoWhileLoop
{
public static void main(String args[])
{
int i=1;
do
{
System.out.println(i);
```

```
i++;
} while(i <=10);
}
}
```

For further reference must go to the link mentioned below:
https://www.javatpoint.com/java-for-loop

Branching Statements:

Break:
It is used to exit from a loop or switch statement. A loop would normally go to completion, but a *break* would cause the early exit

Syntax:
```
class ExampleBreak
{
public static void main(String args[])
{
for(int i = 1 ; i <= 10 ; i++)
{
System.out.println(i);
if(i == 5)
break;
}
System.out.println("After the for loop");
}
}
```

Continue:
If the condition is true, then the *continue* statement skips the current iteration and transfers the control of the loop immediately to the next iteration

Syntax:

```
public class ContinueExample
{
public static void main(String args[])
{
for (int i = 1; i <= 5; i++)
{
```

```
if (i == 2)
{
continue;
}
System.out.println("Value of i ="+ i);
}
}
}
```

Strings in JAVA

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Strings in Java are objects that are backed internally by a char. **String** is a predefined class in the Java library, just like the classes **System** and **Scanner**. The **String** type is non-primitive type.

- **Objects:** Strings are objects, not primitive data types like int or char. They belong to the String class.
- **Immutable:** Once a string is created, its value cannot be changed. Any operation that appears to modify a string actually creates a new string object.
- **Unicode:** Strings in Java use Unicode encoding, supporting a wide range of characters from different languages.
- **Common class:** The String class is one of the most commonly used classes in Java.

Syntax:

<String_Type> <string_variable> = "<sequence_of_string>";

Example:

String str = "oop";

1. **String literals:**

String name = "Alice";
String greeting = "Hello, world!";

2. **Using the new keyword:**

char[] chars = {'J', 'a', 'v', 'a'};
String language = new String(chars);

Common String Operations:

The java.lang.String class provides a lot of methods to work on string. By the help of these methods, operations on string such as trimming, concatenating, converting, comparing, replacing strings etc can be performed.

- **Concatenation:**
    String firstName = "John";
    String lastName = "Doe";

String fullName = firstName + " " + lastName;   // Output: "John Doe"

- **Finding substrings:**
  int index = fullName.indexOf("Doe"); // Output: 5

- **Checking for equality:**
  boolean isEqual = fullName.equals("John Doe"); // Output: true

- **Getting String Length**
 You can use the length() method to return the number of characters in a string.
  Example:
  String message = "Welcome to Java";
  System.out.println("The length of " + message + " is "+ message.length());

- **Getting Characters from a String**
  The s.charAt(index) method can be used to retrieve a specific character in a string **s**, where the index is between **0** and s.length()**–1**.
  For example, message.charAt(0) returns the character **W**, as shown in figure.
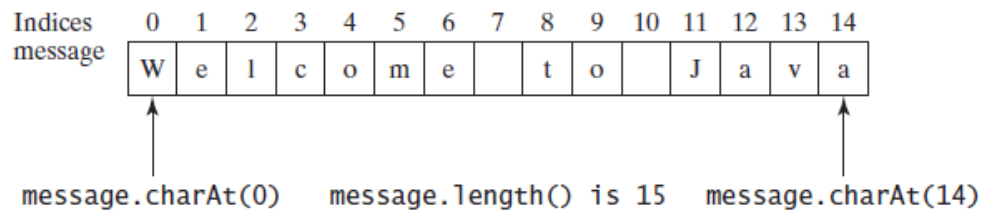
- **Converting Strings**
  The toLowerCase() method returns a new string with all lowercase letters and the toUpperCase() method returns a new string with all uppercase letters.

  *Example***:**
  "Welcome".toLowerCase() returns a new string **welcome**.
  "Welcome".toUpperCase() returns a new string **WELCOME**.

  Example:
  String firstName = "John";
  String lastName = "Doe";
  String fullName = firstName + " " + lastName;   // Output: "John Doe"
  String upper = fullName.toUpperCase(); // Output: "JOHN DOE"
  String lower = fullName.toLowerCase(); // Output: "john doe"

The trim() method returns a new string by eliminating whitespace characters from both ends of the

string. The characters **' '**, **\t**, **\f**, **\r**, or **\n** are known as *whitespace characters*.

Functions used in the exercise

Function	Description	Example	Output
Scanner.nextInt()	Reads an integer input from the user.	int num = scanner.nextInt();	User-entered integer value.
Scanner.next()	Reads a single word as input.	String word = scanner.next();	User-entered word.
Scanner.nextLine()	Reads an entire line of input (including spaces).	String input = scanner.nextLine();	User-entered string with spaces.
Math.max(a, b)	Returns the larger of two values.	Math.max(10, 20)	20
Math.sqrt(x)	Returns the square root of a value.	Math.sqrt(16)	4.0
Math.pow(a, b)	Returns a raised to the power of b.	Math.pow(2, 3)	8.0
String.length()	Returns the number of characters in a string.	"Hello".length()	5
String.toLowerCase()	Convert a string to lowercase.	"HELLO".toLowerCase()	"hello"
StringBuilder.reverse()	Reverse the characters in a string.	new StringBuilder("abc").reverse()	"cba"
String.indexOf(char)	Returns the index of the first occurrence of the specified character in a string.	"Hello".indexOf('e')	1
String.split(regex)	Splits a string into an array of substrings based on the specified delimiter (regex).	"a b c".split("\\s")	["a", "b", "c"]
String.replace(target, replacement)	Replace all occurrences of a target string with a replacement string.	"Hello".replace("l", "x")	"Hexxo"
Character.isLetter(c)	Check if the specified character is a letter.	Character.isLetter('A')	true
StringBuilder	A mutable sequence of characters, often used for string manipulation (e.g., reversing).	new StringBuilder("abc").reverse().toString()	"cba"
String.contains(word)	Checks if a string contains a specific word or substring.	"Hello".contains("He")	true

Exercises

Question 1: (Even or Odd)
Write a Java program that determines whether a given number is even or odd.

Question: 2 (Largest value)
Write a program that takes three values from a user of type float and compares them to find the largest one.

Question: 3 (Vowel/Consonant)
Write a program in Java that takes input from a user and determines whether a given character is a vowel or consonant.

Question: 4 (Factorial)
Write a program that takes input from user and calculates its factorial using loop.

Question: 5 (Fibonacci Series)
Write a loop-based Java program that takes starting and ending values and then prints the Fibonacci series within the specified limit.

Question: 6 (Sum of Digits in number)
Write a program in Java that takes input value from user and print the sum of digits contained in number using a loop.

Question: 7 (Prime)
Write a program that prints the prime numbers within a given range using a loop.

Question: 8 (Triangle Classification)
Write a Java program that takes three integer values representing the sides of a triangle and classifies it as equilateral, isosceles, or scalene triangle.

Question: 9 (Reverse)
Write a Java program that takes input from user and reverse that string.

Question: 10 (Palindrome)
Write a Java program that checks if a given string is a palindrome or not.

Question: 11 (Occurrences of character)
Write a Java program that takes a string and character from the user and counts the occurrences of the given character in a given string.

Question: 12 (Count vowels and consonants)

Write a Java program that takes a string as input and counts the number of vowels and consonants in it.

Question: 13 (Find and Replace)

Write a Java program that takes a sentence, a word to find, and a word to replace it with and print the modified sentence accordingly.

Question: 14 (Count words and characters)

Write a program that takes input as a string from the user and calculates total characters, and words in the string.

Enter any string: My name is Moona.
Total words: 4
Total characters: 17

Question: 15 (Leap year)

Create a program that checks if a year entered by the user is a leap year.