Sukkur Institute of Business Administration University

Department of Computer Science

Object Oriented programming in Java

BS – II (CS/AI/SE)

Spring-2024

Lab # 02: Java Essentials Lab: Exploring type conversion, String and Control Structures

Instructor: Nimra Mughal

Objectives

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

- Some differences of type conversion rules as compared to c++
- Automatic Type Promotion in Expressions
- String Class
- Java Math Class
- Comments in java
- Control Statements

Type conversions rules

Main differences as Compared to C++:

Aspect	C++	Java
Integer Conversions	Allows implicit narrowing conversions	Generally avoids implicit narrowing conversions
Implicit Casting	More implicit casting possibilities	Stricter, relies more on explicit casting
Operator Overloading	Allows operator overloading for custom types	Does not allow operator overloading for primitive types
User Control	Offers more flexibility and control over conversions	Offers explicit casting with less flexibility

Example differences:

1. Narrowing conversion:

C++
int x = 256;
char c = x; // In C++, this silently truncates x to fit in a char (potentially overflows), while Java would trigger an error.

2. Implicit casting:

C++
float f = 3.14;
int sum = f + 1; // In C++, f is implicitly cast to an int without warning, while Java would require explicit casting.

Automatic type promotion in Expressions

When evaluating expressions involving multiple primitive data types, Java automatically promotes smaller data types to larger data types to ensure accurate results and prevent data loss.

- This happens without explicit casting, making code more concise and readable.

Rules:

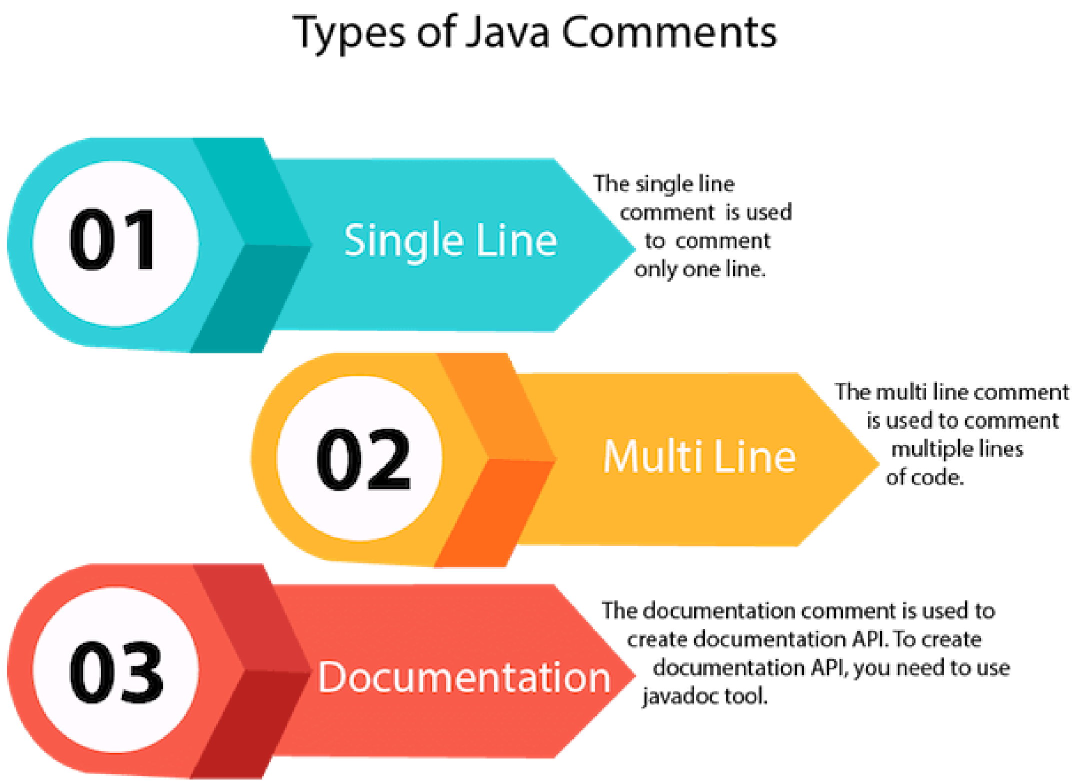
1. byte, short, and char are always promoted to int.
2. If any operand is long, the expression is promoted to long.
3. If any operand is float, the expression is promoted to float.
4. If any operand is double, the expression is promoted to double.

```
// Example 1: byte and short promoted to int
byte b = 10;
short s = 20;
int result = b + s; // b and s are promoted to int before addition
System.out.println(result); // Output: 30
```

```
// Example 2: long promotion
int x = 100;
long y = 200L;
long sum = x + y; // x is promoted to long before addition
System.out.println(sum); // Output: 300

// Example 3: float promotion
int a = 5;
float b = 2.5f;
float result2 = a / b; // a is promoted to float before division
System.out.println(result2); // Output: 2.0
```

Java Comments


Types of Java Comments
01
Single Line
The single line comment is used to comment only one line.
02
Multi Line
The multi line comment is used to comment multiple lines of code.
03
Documentation
The documentation comment is used to create documentation API. To create documentation API, you need to use javadoc tool.

https://www.javatpoint.com/java-comments

Note: Documentation comments will be explored in the next classes. Not Today

Single line	Multi-ilne	Documentation comment
//This is single line comment	/* This is Multiline commet */	/** * *We can use various tags *or heading or author name *We can also use HTML tags * */

Strings in Java

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

- 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.

Creating Strings:

```
<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:

- Concatenation:

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

- **Finding substrings:**

```
int index = fullName.indexOf("Doe"); // Output: 5
```

- **Converting to uppercase/lowercase:**

```
String upper = fullName.toUpperCase(); // Output: "JOHN DOE"
String lower = fullName.toLowerCase(); // Output: "john doe"
```

- **Checking for equality:**

```
boolean isEqual = fullName.equals("John Doe"); // Output: true
```

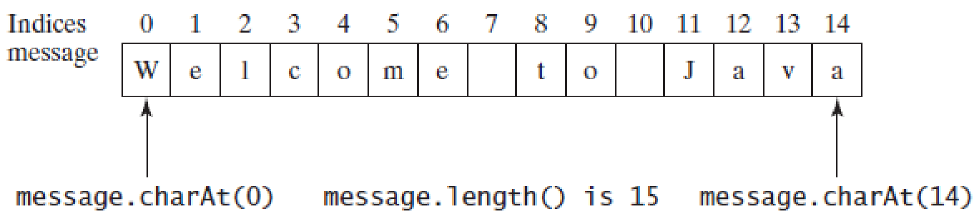
- **Getting String Length**

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**.
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*.

Example code

```java
import java.util.Scanner;

public class StringExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking input from the user
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        // Displaying the input string
        System.out.println("\nInput string: " + inputString);

        // String length
        int length = inputString.length();
        System.out.println("Length of the string: " + length);
        // Converting to uppercase and lowercase
        String upperCaseString = inputString.toUpperCase();
        String lowerCaseString = inputString.toLowerCase();
        System.out.println("Uppercase: " + upperCaseString);
        System.out.println("Lowercase: " + lowerCaseString);

        // Checking if the string contains a substring
        System.out.print("\nEnter a substring to search: ");
        String substring = scanner.nextLine();
        boolean containsSubstring = inputString.contains(substring);
        System.out.println("Does the string contain '" + substring + "'? " +
containsSubstring);

        // Finding the index of a substring
        if (containsSubstring) {
            int index = inputString.indexOf(substring);
            System.out.println("Index of '" + substring + "': " + index);
        }

        // Replacing characters
        System.out.print("\nEnter a character to replace: ");
        char oldChar = scanner.next().charAt(0); // Read first character of input
        System.out.print("Enter a character to replace with: ");
        char newChar = scanner.next().charAt(0); // Read first character of input
        String replacedString = inputString.replace(oldChar, newChar);
        System.out.println("String after replacement: " + replacedString);


        // Splitting the string
        System.out.print("\nEnter a delimiter to split the string: ");
        String delimiter = scanner.next(); // To take string input with no spaces
        String[] parts = inputString.split(delimiter);
        System.out.println("Splitting the string with '" + delimiter + "': ");
        for (String part : parts) {
            System.out.println(part);
        }

        // Substring extraction
        System.out.print("\nEnter start index for substring extraction: ");
        int startIndex = scanner.nextInt();
        System.out.print("Enter end index for substring extraction: ");
        int endIndex = scanner.nextInt();
        String extractedSubstring = inputString.substring(startIndex, endIndex);
        System.out.println("Extracted substring: " + extractedSubstring);
```

```java
        // Checking if the string is empty or null
        System.out.println("\nIs the string empty? " + inputString.isEmpty());
        System.out.println("Is the string null? " + (inputString == null));

        // Closing the scanner
        scanner.close();
    }
}
```

Java Math Class

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

the Java Math Class yourself!

Example code

```java
import java.util.Scanner;

public class MathExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking input from the user
        System.out.print("Enter a number: ");
        double number = scanner.nextDouble();

        // Absolute value
        double absoluteValue = Math.abs(number);
        System.out.println("Absolute value: " + absoluteValue);

        // Square root
        double squareRoot = Math.sqrt(absoluteValue);
        System.out.println("Square root: " + squareRoot);

        // Power
        System.out.print("Enter an exponent: ");
        double exponent = scanner.nextDouble();
        double power = Math.pow(squareRoot, exponent);
        System.out.println("Result of raising " + squareRoot + " to the power of "
+ exponent + ": " + power);

        // Minimum and maximum
        System.out.print("\nEnter another number: ");
        double anotherNumber = scanner.nextDouble();
        double min = Math.min(number, anotherNumber);
        double max = Math.max(number, anotherNumber);
        System.out.println("Minimum of " + number + " and " + anotherNumber + ": "
+ min);
        System.out.println("Maximum of " + number + " and " + anotherNumber + ": "
+ max);

        // Rounding
        double roundedNumber = Math.round(number);
        System.out.println("\nRounded value of " + number + ": " + roundedNumber);

        // Trigonometric functions
        double radians = Math.toRadians(45); // converting degrees to radians
        double sineValue = Math.sin(radians);
        double cosineValue = Math.cos(radians);
        double tangentValue = Math.tan(radians);
        System.out.println("\nSine of 45 degrees: " + sineValue);
        System.out.println("Cosine of 45 degrees: " + cosineValue);
        System.out.println("Tangent of 45 degrees: " + tangentValue);

        // Closing the scanner
        scanner.close();
    }
}
```

Java Control statements

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

Only the highlited ones are new for you. Rest is the history, isin't it? 😉

https://www.javatpoint.com/java-for-loop

Switch with String variable Example

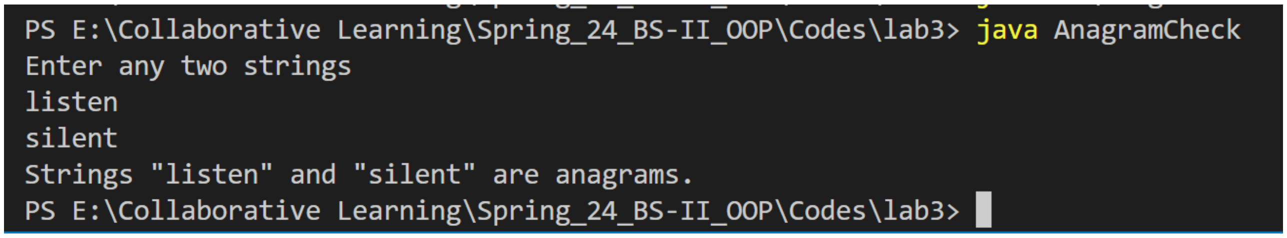
```java
import java.util.*;
public class switchStatement {
public static void main(String[] args) {
    String levelString="Expert";
    Scanner in = new Scanner(System.in);
    levelString = in.nextLine();
    int level=0;
    switch(levelString){
    case "Beginner": level=1;
    break;
    case "Intermediate": level=2;
    break;
    case "Expert": level=3;
    break;
    default: level=0;
    break;
    }
    System.out.println("Your Level is: "+level);
}
}
```

Foreach example

```java
//Java For-each loop example which prints the
//elements of the array
public class ForEachExample {
public static void main(String[] args) {
    //Declaring an array
    int arr[]={12,23,44,56,78};
    //Printing array using for-each loop
    for(int i:arr){
        System.out.println(i);
    }
}
}
```

Exercises

1. **Reverse a string**: Write a java program that reverses the characters of a given string.
2. **Check for palindromes:** Write a java program that determines whether a string is a palindrome (reads the same backward as forward).
3. **Remove duplicate characters**: Write a java program that removes duplicate characters from a string, preserving the order of unique characters.
4. **Find the most frequent character**: Write a java program that finds the character that appears most frequently in a string.
5. **Check for anagrams:** Write a java program that determines whether two strings are anagrams (have the same characters in different arrangements).



```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> java AnagramCheck
Enter any two strings
listen
silent
Strings "listen" and "silent" are anagrams.
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3>
```

6. **Validate email addresses:** Write a java program that checks whether a given string is a valid email address.
7. Write a Java program to **replace each substring of below given sample string**
   a. Sample string : "The quick brown fox jumps over the lazy dog."
   b. In the above string replace all the fox with cat
8. Write a program to print prime number less than 600 and print out the numbers in table format on cmd.
9. Write a program which solves quadratic equations of the form: $ax^2 + bx + c = 0$**. Values of a, b, c can be taken as input from user.**
10. Write a Java program to **round up** the result of variable division