Sukkur Institute of Business Administration University

Department of Computer Science

Object Oriented Programming using Java

BS – II (CS/AI/SE)

Spring-2024

Lab # 04: Unveiling the World of Classes and Objects, Delving into the Depths of String and Array Manipulation

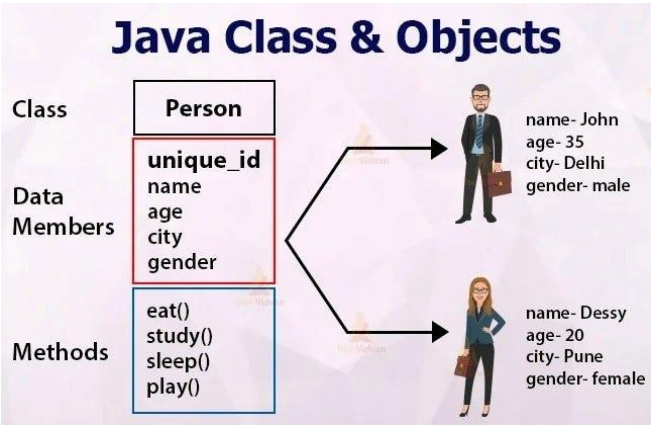
Instructor: Nimra Mughal

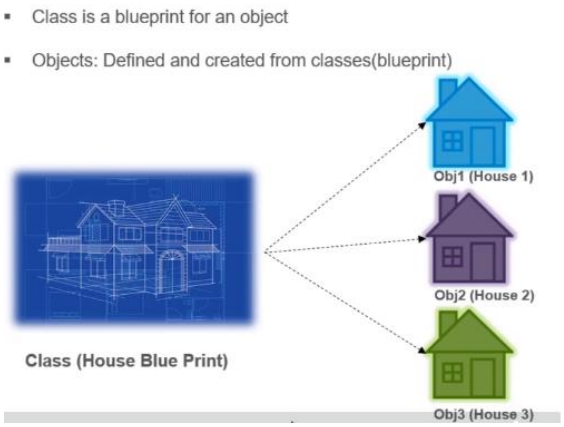
Objectives

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

- Operators' precedence and ternary operator
- Intro to classes and objects
  - Static and Instance variables
  - Methods of the class
- Intro to heap and stack memory
- String advanced
  - Immutable Strings
  - Mutable strings (String Buffer and String builder)
  - String Regular Expressions
- Input ways
  - Scanner
  - JOptionPane
  - Java Command line arguments
  - Buffered reader
- Java Documentation basics
- More practice on arrays

Intro to classes and objects





Static and Instance variables

```
import java.util.Scanner;
class Student {
    static String universityName; // Common attribute for all students
    String name; // Non-static attribute specific to each student
    int id; // Non-static attribute specific to each student
}

public class StaticVariable{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student.universityName = "ABC University"; // Accessing static variable
directly

        // Creating students
        Student student1 = new Student(); //Instantiating Student
        Student student2 = new Student();
        student1.name="Nimra";
        student1.id = 1234;
        student2.name= sc.next();// to take input
        student2.id = sc.nextInt();

        // Output
        System.out.println("Student 1: " + student1.name + ", ID: " + student1.id +
", University: " + student1.universityName);
        System.out.println("Student 2: " + student2.name + ", ID: " + student2.id+
", University: " + Student.universityName);
    }
}
```

Adding Methods to Classes

Implement a **BankAccount class** with attributes accountNumber, accountHolderName, and balance. Include methods to deposit and withdraw funds from the account.

```
public class BankAccount {
    String accountNumber;
    String accountHolderName;
    double balance;
    // Constructor
    BankAccount(String accountNumber, String accountHolderName, double initialBalance) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = initialBalance;
    }

    // Method to deposit funds
    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit of $" + amount + " successful.");
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }
    // Method to withdraw funds
    void withdraw(double amount) {
        if (amount > 0 && balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal of $" + amount + " successful.");
        } else {
            System.out.println("Insufficient funds or invalid withdrawal amount.");
        }
    }
    // Method to display account information
    void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder Name: " + accountHolderName);
        System.out.println("Current Balance: $" + balance);
        System.out.println();

    }
    public static void main(String[] args) {
        // Example usage
        BankAccount account = new BankAccount("123456789", "John Doe", 1000.0);
```

```
        account.displayAccountInfo(); // Display initial account information
        // Deposit $500
        account.deposit(500.0);
        account.displayAccountInfo(); // Display updated account information
        // Withdraw $200
        account.withdraw(200.0);
        account.displayAccountInfo(); // Display updated account information
        // Withdraw $2000 (Insufficient funds)
        account.withdraw(2000.0);
        account.displayAccountInfo(); // Display unchanged account information
    }
}
```

Circle class: Write a class called Circle with attribute radius. Implement methods to calculate the area and circumference of the circle.

```
public class Circle {
    double radius;
    // Constructor, Specialized Methods
    Circle(double radius) {
        this.radius = radius;
    }
    // Method to calculate the area of the circle
    double calculateArea() {
        return Math.PI * radius * radius;
    }
    // Method to calculate the circumference of the circle
    double calculateCircumference() {
        return 2 * Math.PI * radius;
    }

    public static void main(String[] args) {
        // Example usage
        double radius = 5.0;
        Circle circle = new Circle(radius);
        // Calculate and print area
        double area = circle.calculateArea();
        System.out.printf("Area of the circle with radius %.2f: %.2f \n", radius, area);
        // Calculate and print circumference
        double circumference = circle.calculateCircumference();
        System.out.printf("Circumference with radius %.2f: %.2f", radius, circumference);
    }
}
```

Intro to heap and stack memory

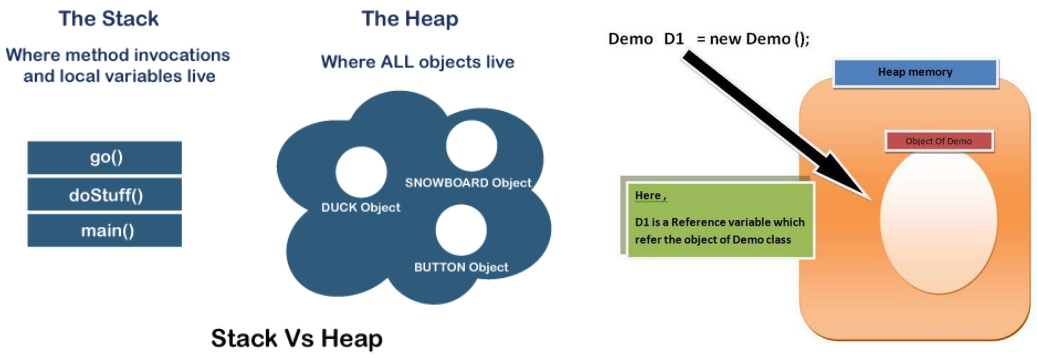
When we talk about **"objects,"** we often mean instances of classes that are referred to using **reference variables.** However, the reference type in C++ is different than the reference type in Java.

Explore More…

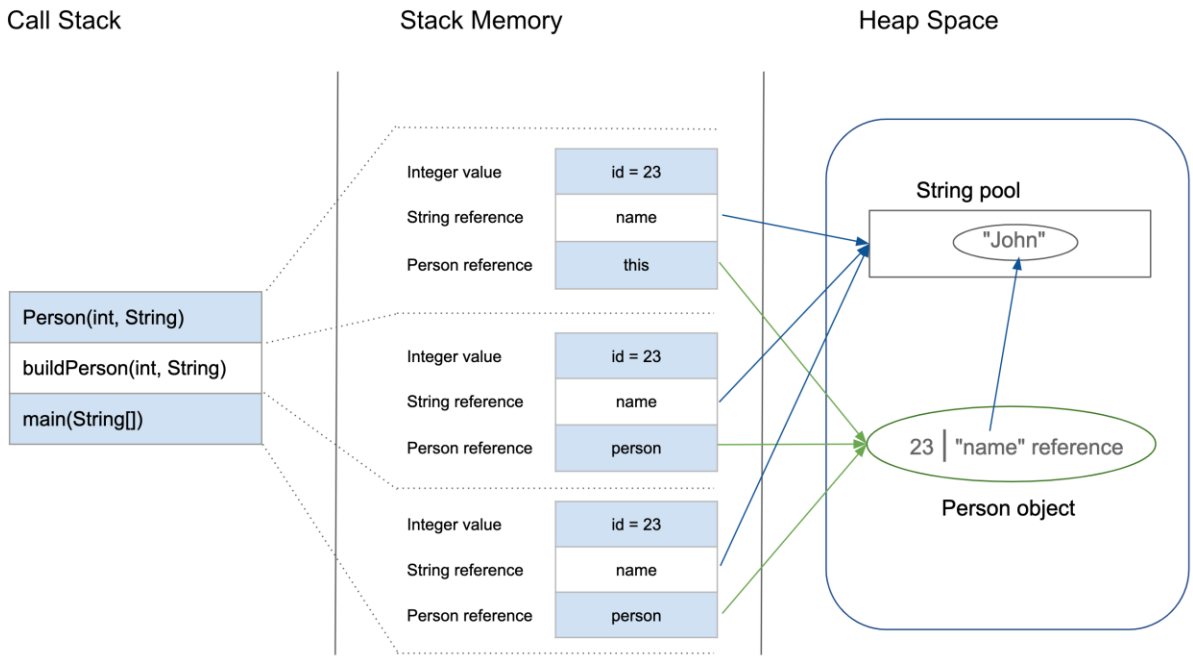
https://www.geeksforgeeks.org/reference-variable-in-java/

https://www.tutorialspoint.com/C-Cplusplus-Pointers-vs-Java-references

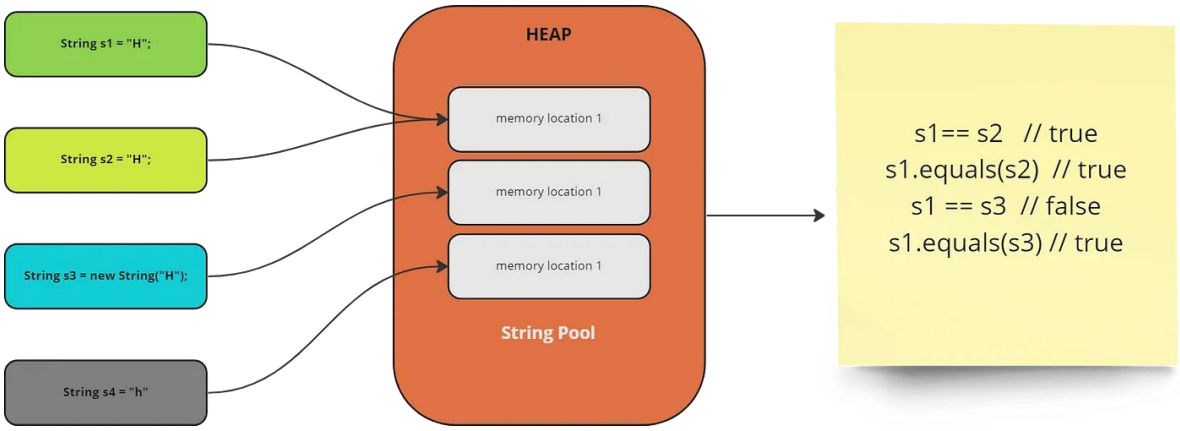
https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/



Stack Vs Heap



String advanced



Immutable

```
/*
This code is created to demonstrate "String pool" in the heap memory. In addition,
it will also demonstrate the Immutable strings
Before Executing This code:
You must be aware of the Heap and Stack memory concept
*/

public class Demo_String {
    public static void main(String args[]){
        String s1 = new String("Nimra");
        String s2 = new String("Nimra");
        System.out.println(s1==s2);//This statement will return False

       //Without the new keyword
        s1 = "Nimra";
        s2= "Nimra";
        System.out.println(s1==s2);//This will return True

        //String are Immutable... Let's see
        System.out.println("Before append: "+ s1 + ": " + s1.hashCode());
        s1 += " Mughal";
        System.out.println("After append: "+ s1 + ": " + s1.hashCode());
        //Hashes are different before and after append

    }
}
```

Mutable strings (String Buffer and String builder)

```
public class Demo_StringBuffer {
    public static void main(String args[]){
        StringBuffer sb = new StringBuffer("Nimra");
        //To append another string/text
        System.out.println("Before append:"+ sb + ": " + sb.hashCode());
        sb.append(" Mughal");
        System.out.println("After append:"+ sb + ": " + sb.hashCode());
        System.out.println(sb.capacity());

    }
}
```

String Regular Expressions

```
import java.util.regex.*;
import java.util.*;
public class RegexExample {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Example 1: Checking if a String contains a specific pattern
        String text1 = "This is the sample text just to exlpore
https://www.finance.gov.pk/";
        String pattern1 = "[a-zA-Z0-9._%+-]+gov.pk";
        boolean containsPattern = Pattern.compile(pattern1).matcher(text1).find();
        System.out.println("Contains 'xyz@gov.pk': " + containsPattern);

        // Example 2: To check whether the email is valid or not
        // Regex pattern for validating email addresses
        String emailPattern = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
        // Input email to be validated
        System.out.println("Enter your email");
        String email = sc.nextLine(); // Replace with the email to be validated
        // Check if the email matches the pattern
        boolean isValidEmail = email.matches(emailPattern);
        System.out.println("The entered email is? :" + isValidEmail);

        // Example 3: Replacing parts of a String using regex
        String text3 = "The quick brown fox jumps over the lazy dog";
        String replacedText = text3.replaceAll("fox", "cat");
        System.out.println("Replaced text: " + replacedText);
    }
}
```

Input ways

JoptionPane

```
import javax.swing.JOptionPane;

public class JOptionPaneExample {
    public static void main(String[] args) {
        //String input(Default)
        String name = JOptionPane.showInputDialog("Enter your name:");
        JOptionPane.showMessageDialog(null, "Hello, " + name + "! Welcome.");

        //Integer Input
        String str = JOptionPane.showInputDialog("Enter Any Number: ");
        int num = Integer.parseInt(str);
        JOptionPane.showMessageDialog(null, "you entered" + num + "! Welcome.");

        //For float/double
        str = JOptionPane.showInputDialog("Enter Any Number: ");
        float num1 = Float.parseFloat(str);
        JOptionPane.showMessageDialog(null, "you entered" + num1 + "! Welcome.");

    }
}
```

Command Line Arguments

```
public class CMDArgumentsExample {
    public static void main(String[] args) {
        if(args.length > 0) {
            System.out.println("Arguments passed from command line:");
            for (String arg : args) {
                System.out.println(arg);
            }
        } else {
            System.out.println("No arguments provided.");
        }
    }
}
```

To run this code after compilation

```
java CMDArgumentsExample argument1 argument2 argument3
```

Output formatting

```
public class PrintfExample {
    public static void main(String[] args) {
        String name = "John";
        int age = 30;
        double height = 6.1;

        System.out.printf("Name: %s, Age: %d, Height: %.1f\n", name, age, height);
        System.out.println();
        // Format strings to be placed in fixed width
        System.out.printf("Name: %-10s, Age: %-5d, Height: %-5.1f\n", name, age,
height);

        double d = 4.5;

        System.out.println(d);
        // Shows 2 digits after the decimal point
        System.out.println(String.format("floatValue: %.2f", d)); // Shows 2 digits
after the decimal point

    }
}
```

Arrays in java

https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

https://www.hackerrank.com/challenges/java-1d-array-introduction/problem?isFullScreen=true

https://www.w3resource.com/java-exercises/array/index.php

Operators Precedence and Ternary Operator

https://www.javatpoint.com/operators-in-java

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

https://www.refreshjava.com/java/operator-precedence

https://www.geeksforgeeks.org/java-ternary-operator-with-examples/

Java Documentation Comments

```java
import java.io.*;
/**
* Add Two Numbers!
* The AddNum program implements an application that
* simply adds two given integer numbers and Prints
* the output on the screen.
* <p>
* <b>Note:</b> Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
public class Account {

   /**
   * This is the main method which makes use of addNum method.
   * @param args Unused.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[]) throws IOException {
      AddNum obj = new AddNum();
      int sum = obj.addNum(10, 20);

      System.out.println("Sum of 10 and 20 is :" + sum);
   }
}
```

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javac DocExample.java
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javadoc DocExample.java
Loading source file DocExample.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 21.0.2+13-LTS-58
Building tree for all the packages and classes...
Generating .\DocExample.html...
```

The javadoc tool recognizes the following tags:

https://www.tutorialspoint.com/java/java_documentation.htm

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

Exercises

1. **Rectangle class:** Write a Java class called Rectangle with attributes **width and height**. Include methods to calculate the **area and perimeter** of the rectangle.

   Create two instances/ reference variables of the Rectangle class and:

   - **Set Object's instance variables using Scanner**: set the values of the instance variables using the Scanner class.
   - **Set Object's instance variables using JOptionPane**: set the values instance variables using JOptionPane.
   - **Call the Methods**: Call both methods of area and perimeter and display the output.

2. **An array of Objects:** In the practice exercise, you've already created a Student class and instantiated two objects of that class. Now, I'd like you to perform the following tasks:
   - **An array of Student Objects:** Create an array of **Student Class** with a size of 5.
   - **Setting Object Variables using Scanner**: Use a loop to set the values of the object variables in the array using the Scanner class.

3. **Mega Exercise: Solve Exercises from the HackerRank from the covered topics (As many as you can. I have listed here two basic challenges)**

https://www.hackerrank.com/domains/java
https://www.hackerrank.com/challenges/java-output-formatting?isFullScreen=true
https://www.hackerrank.com/challenges/java-1d-array-introduction/problem?isFullScreen=true

4. **Generate random passwords:** Write a code that generates a random password of a specified length, using a combination of letters, numbers, and symbols.

5. **Create a class Person with fields for name, age, and address.**
   - Use @author and @version tags in the class comment.
   - Format comments with HTML tags for emphasis and lists (e.g., <b>).

6. **Check for anagrams:** Write a code 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>
```

7. **Password Validator:** Write a Java code that can verify whether the string is a valid password or not
    a. Minimum 8 characters long
    b. Contains at least one digit
    c. Contains at least one alphabet
    d. Contains at least one special character

8. Solve Exercises of **Bitwise operators** from previous labs if you couldn't solve them earlier