Eclipse

Session 9

Project 1

Data Type and Variable Scope

Lesson Objectives

In this lesson you will learn to:

- The different of variable scope
- Apply casting in Java code

Data Type

In Java, data types:

– Are used to define the kind of data that can be stored inside a variable.

– Ensure that only correct data is stored.

– Are either declared or inferred.

Type	Keyword	Description
Boolean	bool	Stores either value true or false.
Character	char	Typically a single octet(one byte).
Integer	int	The most natural size of integer for the machine.
Floating point	float	A single-precision floating point value.
Double floating point	double	A double-precision floating point value.

Casting Data

A cast signals a type conversion. If you cast one type as another type, you are explicitly converting the item from it's original type to the type that you specify. A cast does not round a decimal number, but instead, will truncate it. An example of casting a double as an int is:

```
double x = 5.745
int y = (int)x;  //y is now equal to 5
```

Let's Code

- Create TesCasting Project, then Create tesCasting package
- Create CastingData class
- Add code to class, then Compile and Run the program

```
public class castingData {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        byte a = 50;
        int b = 5;
        float c = 2;

        int d = (a*b/c);
        int e = (a*b)/251;

        System.out.println("a = " + a +" b = "+ b +" c = "+ c +" d = "+ d);
        System.out.println("e = "+e);

    }

}
```

- Add Casting data type of **d** and **e** variable, as shown bellow

```
int d = (int)(a*b/c);
int e = (int)((a*b)/251); //hasil operasi di casting ke int
```

Variable Scope

- Scope is used to describe the block of code where a variable exists in a program.
- It is possible for multiple variables with the same name to exist in a Java program.
  - In most cases, the innermost variable has precedence.
- A variable exists only inside the code block in which it is declared.
- Once the final brace of the block } is reached:
  - The variable goes out of scope.
  - It is no longer recognized as a declared variable

The scope of a variable refers to where is a variable visible or accesible.

- Global variables are declared outside any function, and they can be accessed (used) on any function in the program.

- Local variables are declared inside a function, and can be used only inside that function. It is possible to have local variables with the same name in different functions. Even the name is the same, they are not the same. It's like two people with the same name. Even the name is the same, the persons are not.

Example Local Variable

```
public void someMethod()
   {
     if(gameOver && score>highScore)
     {
       String name;                                    //Point A
       System.out.println("Please enter your name:");
         name=reader.next();
     }                                                 //Point B

     System.out.println("Thank you " + name + ", ");
     System.out.println("your high score has been saved.");
   }
```

Example Global Variable

```
public void someMethod()
  {
    String name;                          //Point A
    if(gameOver && score>highScore)
    {
      System.out.println("Please enter your name:");
      name=reader.next();
    }                                     //Point B
    System.out.println("Thank you " + name + ", ");
    System.out.println("your high score has been saved.");
  }
```

PROJECT 2

STRING

Lesson Objectives

In this lesson you will learn to:

- Instantiate (create) a String
- Describe what happens when a String is modified
- Use the + and += operators for concatenating Strings
- Interpret escape sequences in String literals

What is string ?

- A String is an object that contains a sequence of characters. Declaring and instantiating a String is much like any other object variable.

- However, there are differences:

– They can be instantiated (created) without using the new keyword.
– They are immutable.

- Each String Has an Index

Since Strings are a representation of a sequence of characters, each character of a String has an index.

Method in String Class

- length()
  Returns the length of this string.

- toUpperCase()
  Converts all of the characters in this String to upper case

- toLowerCase()
  Converts all of the characters in this String to lower case

- contains(CharSequence s)
  Returns true if and only if this string contains the specified sequence of char values.

Method in String

- concat(**String str**)
  Concatenates the specified string to the end of this string.

- indexOf(**char c)**
  Returns the index value of the first occurrence of c in string.

- charAt(**int index)**
  Returns the character of the String located at the index passed as the parameter

- substring(**int beginIndex, int endIndex**)
  Returns a new string that is a substring of this string

- Create TesString Project, then Create tesString package
- Create StringOperations class
- Add code to class, then Compile and Run the program

```
public class StringOperations{
     public static void main(String[] args){
       String string1 = "Hello";
       String string2="Lisa";
       String string3="";                    //empty String or null
       string3="How are you ".concat(string2);
       System.out.println("string3: "+ string3);
       //get length
       System.out.println("Length: "+ string1.length();
       //get substring beginning with character 0, up to, but not
       //including character 5
       System.out.println("Sub: "+ string3.substring(0,5));
       //uppercase
       System.out.println("Upper: "+string3.toUpperCase());
     }
}
```

Method in String

Add this code to StringOperation Class

```
String string4 = "How are you "+ string2;
System.out.println("string4 : "+ string4);
System.out.println("string : "+ (string1 += string2));
System.out.println(string2.indexOf('a'));
System.out.println(string2.charAt(2));
```

Method in String Class

- compareTo(string AnotherString)

Compares two strings lexicographically.

```
String s1 = "abc";
String s2 = "cde";
System.out.println(s1.compareTo(s2));
```

- Equals()

Compares this string to the specified object.

```
String s1 = "abc";
String s2 = "ABC";
System.out.println(s1.equals(s2));
```

Save Your Project

Project 3

Control Statement

Lesson Objectives

In this lesson you will learn to:

- Use if-else logic and statements
- Apply switch logic and statements in Java code
- Use break and default effectively in a switch statement
- Use the ternary operator

If – Else Statement

- To build an if-else statement, remember the following rules:
  - An if-else statement needs a condition or method that is tested for true/false. For example:

```
if(x==5)

if(y >- 17)

if(s1.equals(s2))
```

```
import java.util.Scanner;
public class Calculator{
       public static void main(String[] args){
  Scanner in = new Scanner(System.in);
  int answer = 0;
  System.out.println("Enter a number:");
  int num1 = in.nextInt();
  System.out.println("Enter another number:");
  int num2 = in.nextInt();
  System.out.println("Enter the operand:");
  char input = in.next().charAt(0);
  if( input == '*' )
      answer = num1 * num2;
  else if( input == '/' )
      answer = num1 / num2;
  else if( input == '%' )
      answer = num1%num2;
  else if( input == '+' )
      answer = num1 + num2;
  else if( input == '-' )
      answer = num1 - num2;
  else
      System.out.println("Invalid Command");
  System.out.println("The result is: " + answer);
}
```

Switch Statement

Like the if-else example earlier, consider a program that takes two integer inputs from a user and performs a specified mathematical operation.

A switch statement is another way of changing program flow depending on the input value.

A switch statement uses 3 keywords: switch, case, and default.
– **switch**: specifies which variable to test for value.
– **case**: compares the value of the switch variable.
– **default**: when the input does not match any of the cases, the compiler chooses the default action (like else in a list of if statements).

```java
import java.util.Scanner;
public class Calculator{
          public static void main(String[] args){
              Scanner in = new Scanner(System.in);
              int answer = 0;
              System.out.println("Enter a number:");
              int num1 = in.nextInt();
              System.out.println("Enter another number:");
              int num2 = in.nextInt();
              System.out.println("Enter the operand:");
              char input = in.next().charAt(0);
              switch (input){
                 case '*' :
                    answer = num1 * num2;
                    break;
                 case '/' :
                    answer = num1 / num2;
                    break;
                 case '%' :
                    answer = num1 % num2;
                    break;
                 case '+' :
                    answer = num1 + num2;
                    break;
                case '-' :
                    answer = num1 - num2;
                    break;
                default:
                    System.out.println("Invalid Command.");
              }
              System.out.println("The result is: " + answer);
   }
}
```

Ternary Operator

The ternary operator (? :) in Java is used to create a shorter version of an if-else statement.

- Here, an if-else statement is used to check for String equality

```
String s1 = "Hello";
String s2 = "Goodbye";
if(s1.equals(s2))
        System.out.println("Yes");
else
        System.out.println("No");
```

- A similar result is achieved using the ternary operator

```
String s1 = "Hello";
String s2 = "Goodbye";
String answer = s1.equals(s2) ? "Yes" : "No";
System.out.println(answer);
```

Example

```
int c=3, b=2;
System.out.println("min = " + (c < b ? c : b));

a = -10;
int absValue = (a < 0) ? -a : a;
System.out.println("abs = " + absValue);

float result = true ? 1.0f : 2.0f;
System.out.println("float = " + result);

String s = false ? "Dude, that was true" : "Sorry Dude, it's false";
System.out.println(s);
```

Let's Code Program

- Create project and package name as ControlStatementProject
- Create leapYearCalculator Class
- Code the program into the following :
  - Program will receive input month (1-12) and year
  - Use Ternary Operator to determine is the inputted month even or odd
  - Use Switch-Case to determine numbers of day is it 30 or 31 from the inputted month. If the inputted month is more than 12 and less than 1, the program will give an "Invalid input".
  - If the inputted year is a leap year, use If-Else to determine is the numbers of day is 28 or 29.
  - We can calculate the leap year if the inputted year divisible by 4 and not divisible by 100, or divisible by 400

Save Your Project